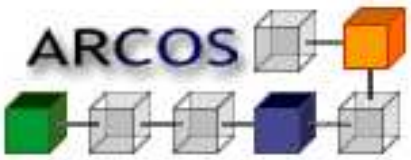
Universidad
Carlos III de Madrid

Grado de Ingeniería en Informática

Leganés, Septiembre de 2015

TRABAJO FIN DE GRADO

Estudio sobre el diseño e implementación de un sistema de almacenamiento utilizando los servicios distribuidos dados por Apache ZooKeeper



Autor: Teodor Sergeev Totev
Tutor: Alejandro Calderón Mateos

Agradecimientos

Querría agradecer a mi madre y José por el apoyo que me han dado durante estos 4 años de carrera, y por no dejar de empujarme siempre para avanzar.

Además no quiero olvidarme de de todo estos compañeros que he tenido durante estos 4 años de carrera y toda la buena gente que he podido conocer, compartiendo horas de trabajo y de clase. Como no en especial Omar y Diego, que más que compañeros son amigos.

Obviamente también tengo que agradecer a todos los profesores que me han ayudado a formarme como informático, en especial Alejandro Calderón y Carlos Linares, pero no quisiera olvidarme de Félix García o Ana Isabel Gonzáles y entro otros muchos nombres.

También a mis amigos de siempre, que han estado a su manera apoyándome y a veces desde la distancia.

Y en especial querría agradecer a María Herrero Marín, que siempre has estado a mi lado, en los momentos malos y los buenos, creyendo en mí incluso cuando yo no lo podía hacer. Haber empezado esta aventura y haber llegado hasta aquí es en gran parte gracias a ti.

Resumen

En este documento se hablará sobre el planteamiento y diseño de mi TFG, así como del desarrollo del primer prototipo.

Se trata de crear un sistema que sea capaz de centralizar y sincronizar los ficheros de un usuario. Estos ficheros pueden estar tanto en servidores de la nube, como en servidores propios del usuario o incluso terminales como PC o un smartphone. El sistema tiene que tener la capacidad de localizar y saber donde están todos los ficheros que el usuario a añadido a su cuenta particular, de forma que se los enseñe a él todos juntos, pero cuando pida un fichero, saber donde buscarlo y proporcionárselo al usuario de una forma transparente. Además el sistema también tiene que ser activo y tener funcionalidades que el usuario no tenga que realizar a mano, como por ejemplo crear replicas de un fichero según qué condiciones o detectar cuando alguno ha sido modificado o borrado directamente desde el servidor en el que se localizaba, entre otros.

Para realizar este proyecto y como con casi todos, hay que hacer un estudio de la competencia que existe hoy en día y las alternativas o posibles soluciones que hay al problema planteado. El estudio se centra en 3 alternativas principales, que son: El almacenamiento en nube, herramientas o servicios de unificación de cuentas en la nube y por ultimo el almacenamiento NAS. También se estudian las tecnologías de las que se puede hacer uso para el desarrollo para los diferentes aspectos, como: el almacenamiento de los meta-datos, el de los datos, el servidor de aplicaciones, o los lenguajes de programación utilizados.

Luego ya se podría plantear el análisis de casos de uso y requisitos, así como el diseño del proyecto, el cual parte por una vista general del mismo, las posibles alternativas y el planteamiento detallado de la alternativa escogida.

Hay que hablar, además, de la implementación de un primer prototipo, sobre el que se realizan algunas pruebas para determinar la viabilidad del proyecto y saber si se puede llevar a cabo o si realmente podría cubrir las necesidades u objetivos marcados.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 Introducción

1.1 Motivación

A lo largo de mi grado me di cuenta de un problema que me ocurría tanto a mí como a mis compañeros. Dicho problema se producía sobretodo a la hora de realizar prácticas. Cuando nos poníamos a desarrollar prácticas nos dábamos cuenta que teníamos partes y versiones de la misma práctica en DropBox, en Google Drive, en el PC personal de casa de cada uno, en el PC de la universidad, etc. y eso de una misma práctica. Si consideramos todas las prácticas que hacíamos de todas las asignaturas, acabamos teniendo material dispersado por un sin fin de nubes, terminales, incluso a veces te encontrabas enunciados o apuntes en los smartphones. Por esta razón y por el interés que tengo sobre la metería decidí, como TFG, realizar algo que pudiera facilitar toda esta tarea de saber donde tienes todo y poder acceder de una forma más sencilla a todo el material que puedes tener en distintos sitios. Para ello habría que diseñar una plataforma que gestionara ficheros de forma distribuida y que además pudiera conectarse a todas las fuentes de almacenamiento que el usuario fuera a usar, dirigida al mayor número de usuarios y al mayor tipo de usuario, tanto usuarios casuales, como profesionales.

Una vez planteado esto el mayor interés vino de hacer algo diferente a lo que propone hoy en día uno de los sistemas de mayor crecimiento en cuanto a almacenamiento de datos, que es el almacenamiento en nube. Uno de los problemas viene precisamente de este tipo de almacenamiento, pero a su vez ofrece muchas cosas buenas, así que la intención es buscar un término medio entre las bondades que ofrece el almacenamiento en nube mientras que se consiguiera corregir sus carencias. Para ello se pensó en un sistema de almacenamiento vitaminizado, para que además de gestionar los almacenamientos en nube, ofrezca una gestión también de las fuentes de almacenamiento propias del usuario, en una especie de fusión entre los dos tipos de almacenamiento, en nube y nube personal.

El proyecto, en gran parte, se basa en los dos conceptos siguientes:

- Permita al usuario/cliente centralizar y unificar todas sus fuentes de almacenamiento, pudiendo incluso añadir servidores propios o terminales de una forma transparente para el mismo

- Y que por otro lado este sistema sea activo y no pasivo, ofreciendo al cliente una inteligencia basada principalmente en la definición de reglas entre otras funciones.

¿Qué quiere decir centralizar y unificar?

Lo que se pretende es que el usuario pueda usar nuestro sistema o como un almacenamiento en nube al uso o que el mismo usuario pueda añadir y asociar a su cuenta otros tipos de almacenamientos, ya sean otras cuentas de almacenamiento en

nube como Dropbox y Google Drive, servidores propios o incluso terminales como PC's, Smartphones, etc. y tener toda esta información centralizada y transparente de cara al usuario. De forma que el usuario pida al sistema un fichero y el mismo sistema se encargue de saber donde buscar el fichero y entregárselo al usuario. De la misma forma el sistema debería encargarse de sincronizar y actualizar la información en caso de que se produzca algún cambio en alguno de los ficheros, de forma directa desde los servidores asociados, de los que esté tenga conocimiento o "conciencia".

¿Sistema activo?

De lo que se trata es que el sistema realice acciones de forma transparente para el usuario en la gestión de los datos, además de que el usuario pueda gestionar y crear sus propias reglas y programarlas al gusto de cada uno. Por ejemplo, que en caso de que un fichero se convierta en fichero "hot" (que se accede un número determinado de veces) esté se consideré como un fichero importante y automáticamente se haga del mismo varias replicas en los servidores que el usuario tenga o en el mismo sistema, dependiendo de cómo este definida la regla. La definición de reglas es un aspecto interesante que hace que nuestro sistema sea activo y haga cosas por nosotros, pero de forma complementaria el sistema también tiene que tener la capacidad, como se dijo anteriormente, de actualizar su información en caso de que se produzca algún cambio en los ficheros o realizar algunas actividades que no están al alcance de la gestión del usuario.

1.2 Objetivos

Vamos a definir algunos de los principales objetivos que tiene que tener nuestro sistema para que pueda ser de utilidad a nuestros usuarios y que se intentarán cumplir en el desarrollo de este proyecto para que realmente pueda ayudar al día a día de los mismos.

- El usuario tiene que tener la capacidad de tener su propia cuenta y poder gestionar la misma, al menos con las acciones más básicas, como pueden ser acceder a la misma, abandonarla, cerrarla. En futuras versiones del prototipo también se le ofrecerá la posibilidad de cambiar información de la misma, como por ejemplo, su contraseña (está opción no estará disponible en el prototipo (5.2)).

- Además de eso el usuario tiene que ser capaz de añadir, recuperar o borrar, así como visualizar todos los ficheros que tenga en su cuenta. Tanto la opción de cifrar, como la de gestionar servidores que pueda tener en su cuenta, quedarían pendientes como trabajos futuros (está opción no estará disponible en el prototipo (5.2)).

- El servicio tiene que ofrecer una vista transparente para el usuario, de forma que el usuario no se tenga que preocupar de donde esta cada fichero si no que el mismo sistema se encargue de la localización y solicitud del mismo y de ofrecérselo al usuario.

- Los usuarios deberán usar la web para acceder, pero en versiones posteriores se podrá usar también la aplicación o la interfaz REST para que puedan crear su propia forma de acceso y gestión del sistema (8.3.2.1).

- Los usuarios tienen que ser capaces de tener la futura posibilidad (8.3.1) de crear grupos de usuarios, los cuales pueden compartir ficheros, con el permiso del dueño, o servidores enteros con acceso a la información que contengan.

- Crear un prototipo inicial por el cual podemos empezar a tratar el sistema y comprobar su alcance inicial, funcionamiento básico y comprobar su viabilidad inicial.

1.3 Estructura de la memoria

En este apartado vamos a proceder a describir la estructura de la memoria de forma orientativa y para ayudar al lector al buen seguimiento de la lectura de la misma. Para ello vamos a describir el contenido de cada capitulo principal y los anexos de forma concisa.

- Capitulo 1 **Introducción**: Es el capitulo en el que nos encontramos y en el que sirve de introducción, tanto al proyecto en cuestión como al documento que lo complementa.

- Capitulo 2 **Estado del arte**: Este capitulo es en el que se podrá ver lo que ofrece la competencia relacionado con mi proyecto, además de algunas de las posibles soluciones que se podrían acercar al problema que me concierne, para terminar relacionando todas las opciones entre sí. Además también haré un estudio de las tecnologías relacionadas para completar el trabajo, algunas de las posibles alternativas existentes en cada caso y las pertinentes comparativas entre ellas.

- Capitulo 3 **Análisis del problema**: Este es el capitulo perteneciente al análisis del problema, para ello se realizará un estudio de requisitos, exponiendo los mismos en tablas. Además también se hará un estudio de los casos de usos posibles sobre el sistema para finalizar relacionando los requisitos y los casos de usos en una matriz de trazabilidad.

- Capitulo 4 **Diseño de la solución técnica**: Aquí vamos a ver uno de los capitulo de más peso, relacionado con el diseño del proyecto. Para empezar, y en cierto modo, como una forma de introducción se realizará un diseño general sobre el proyecto, para luego continuar con las alternativas de diseño que se plantean. Una vez realizado dicho estudio de las alternativas y justificada la elección de una de ellas se procederá a desarrollar en detalle el diseño escogido.

- Capitulo 5 **Implementación e implantación**: Después de realizar el diseño se procederá a definir la implementación e implantación del sistema. Para ello se realizará definición del mismo. Además de eso se indicará de donde descargar el

software necesario, así como su correspondiente configuración para poder poner en marcha el sistema.

- Capitulo 6 **Pruebas de aceptación**: En este capitulo se realizará un estudio de los resultados del prototipo, así como de la viabilidad del proyecto. Para ello se realizará una batería de pruebas de aceptación y el análisis de las mismas pruebas.

- Capitulo 7 **Planificación y presupuesto**: En este capitulo se verá la planificación realizada para el proyecto, así como el presupuesto que se necesitaría para el desarrollo, en un principio, del análisis, diseño, prototipo, etc.

- Capitulo 8 **Conclusiones y trabajos futuros**: Aquí se hablará sobre las conclusiones extraídas de realizar el proyecto y el prototipo, de sus desarrollos, así como de los aspectos personales. También se verán los trabajos futuros tanto sobre el modelo como sobre el prototipo.

- Capitulo 9 **Bibliografía**: Este capitulo contiene la bibliografía recopilada para el desarrollo y estudio del proyecto.

- Anexo 1 **Ingles**: Este es el anexo que contiene el contenido exigido en ingles, el cual incluye la introducción, las conclusiones, así como el resumen del documento.

2 Estado del arte

2.1 Introducción

Como en todos los desarrollos siempre es recomendable empezar por ver que es lo que hay desarrollado hasta el momento y que es lo que ofrece cada opción que ya existe para poder comprobar hasta que punto estas aplicaciones y servicios existentes cubren nuestras necesidades.

2.2 Posibles soluciones existentes

Para empezar, hay que tener en cuenta que nuestro servicio se basa en el almacenamiento distribuido, para lo cual existen ya algunas opciones, como en el caso de cliente-servidor, como el FTP (*File Transfer Protocol*) o por ejemplo la transferencia de ficheros basada en el P2P (P*eer-to-peer*). Estas formas de transferir ficheros sirven como base para el desarrollo de otro tipo se servicios más completos y más simples de usar que han ido evolucionando a lo largo del tiempo. Uno de estos tipos de servicios es la nube, un sistema de almacenamiento que hoy en día es muy extendido y todavía está en proceso de crecimiento. Su utilidad máxima viene de no requerir un espacio de almacenamiento al usuario, ofreciéndoselo el mismo servicio, así como el hecho de tener toda su información en un mismo sitio. El problema es que ese mismo crecimiento y expansión del almacenamiento en nube, también produce uno de los grandes problemas que hay sobre este tipo de almacenamiento, que es el gran número de cuantas que podemos llegar a tener ya sea de una misma empresa o de varias, por lo que acabamos perdiendo la noción sobre nuestros propios ficheros, su localización, replicas, etc.

Para solventar este tipo de problema existen unos servicios, por lo general desarrollados para la web, pero se han seguido ampliando y desarrollando aplicaciones y también app’s para los smartphones, que nos ofrecen la posibilidad de tener nuestras cuentas de la nube en una misma vista, y tener localizada toda nuestra información. Profundizaremos más en las principales plataformas, pero uno de sus principales problemas es que solo sirven para centralizar la información de la nube y que las opciones que ofrecen algunas de ellas cubren solo “necesidades” muy básicas.

También existe la posibilidad del almacenamiento NAS, para el cual aquí se analizará el de pago, puesto que el resto está en desarrollo o no ofrece una estabilidad buena. Este tipo de almacenamiento se considera como la “nube personal”, usando un almacenamiento que ya tiene su software preinstalado, el cual permite la posibilidad de acceso y sincronización de varios terminales.

2.2.1 La nube

Para este estudio se han examinado en concreto DropBox y Google Drive. Ambas ofrecen una variedad de opciones en cuanto a gestión y capacidad, según lo que el usuario busque. Existen unas cuentas "básicas" las cuales se ofrecen nada más registrarse, ambas con las acciones justas para una gestión básica como las de borrar, añadir ficheros o carpetas, crear carpetas compartidas, etc. y también con un espacio limitado. También ambos casos ofrecen otros packs por una determinada cuota las cuales, por lo general, te ofrecen más seguridad, mejores herramientas de gestión y espacio de almacenamiento. Además de eso ambas tiene aplicaciones y app's para poder sincronizar los datos y trabajar de forma directa con la nube.

Todo eso esta bien, pero como todas las plataformas de almacenamiento en nube se queda en eso, un almacenamiento pasivo el cual hay que gestionar de forma manual y que además solo te ofrece el almacenamiento interno de la empresa en cuestión en sus servidores de almacenamiento, sin posibilidad de añadir ningún otro tipo de espacio de almacenamiento externo.

2.2.1.1 DropBox

Se trata de una de las nubes más importantes del mercado y de las que más extendida está entre los usuarios [1].

- Sus aplicaciones son bastante robustas y estables, ofreciendo una buena sincronización de los datos del usuario.

- La web del servicio es bastante simple, lo que ayuda bastante a los nuevos usuarios o a los usuarios más casuales, pero por otro lado puede restar capacidades para los usuarios más avanzados.

- Ofrece una cuenta gratuita al registrarse con un espacio de almacenamiento de 2GB, quizás algo justo, pero ofrece formas gratuitas de ampliación del mismo.

- Tiene versiones de pago que ofrecen a los usuarios opciones más extendidas, así como un espacio de almacenamiento mayor.

- Tiene la ventaja de que sigue siendo el servicio en nube más común y sobretodo a la hora de compartir ficheros o iniciarse en el mundo de la nube (también dada su web sencilla) cuenta con esa ventaja.

2.2.1.2 Google Drive

Este es otro de los servicios de nube más populares que existen. Tiene algunas ventajas frente a DropBox, pero también algunas inconveniencias [2].

- Ofrecer un espacio mayor, en este caso que DropBox, para las cuentas gratuitas, en concreto empieza por 15 GB pero con posibilidad de ir aumentándolo, pero en contra tiene que comparte su espacio con todas las demás cuentas que te crea, en caso de que se las de uso, como Gmail, Google +, etc.

- Además ofrece la posibilidad de editar documentos Google (e.j. Google Docs) de forma simultánea por varios usuarios, lo que permite que los mismos usuarios tengan la capacidad de sincronizar o autentificar sus trabajos en tiempo "real", puesto que todos los usuarios que estén editando el documento pueden ver lo que van haciendo los demás. Este es quizás uno de los mayores atractivos en detrimento a la otra opción analizada.

- Google Drive también ofrece aplicaciones y app's para el acceso a su servicio, pero estos a veces ofrecen algunos problemas en la sincronización, por lo que no son tan sólidas.

- Obviamente también tiene versiones de pago, que ofrecen más espacio de almacenamiento, así como algunas opciones y gestiones complementarias.

- En algunos casos puede ser un problema, que al crearte una cuenta en Google Drive internamente Google te crea cuentas para todos sus servicios, cierto es que no las activa, pero están allí, aún que a algunas personas no les importe o incluso les guste, a nivel personal es algo que no me gusta mucho.

2.2.2 Servicios de unificación de la nube

El nivel del almacenamiento en nube, en este análisis, lo podemos definir como un nivel base, ya que forma una parte de nuestro proyecto final de unificar la información y espacios de almacenamiento que podemos tener. Con este fin vamos a analizar las siguientes herramientas que existen hoy en día, que se dedican a unificar las diferentes cuentas en la nube que tenemos y ofrecernos una interfaz única sobre el contenido que tenemos en todas las cuentas. Por lo general se trata de servicios ofrecidos mediante la web, pero se han seguido ampliando y desarrollando aplicaciones y también app's para los smartphones. Estos servicios de cierta manera tendrían un acercamiento a nuestra idea de centralizar y dar transparencia al usuario en cuanto al contenido. El problema es que al igual que las nubes en si, su gestión sigue siendo estática y limitada al almacenamiento que te ofrecen las plataformas de la nube, además de que las mismas solo te ofrezcan eso, poder añadir cuentas de almacenamiento en nube y ningún otro tipo de servicio o almacenamiento externo a estas. Además por lo general suelen gestionar solo 1 cuenta por cada empresa de almacenamiento en nube o varias mediante un cuenta Premium, aún que no siempre sea ese el caso.

A continuación vamos a ver algunas de las aplicaciones más importantes de este sector.

2.2.2.1 Jumptuit

Esta herramienta, principalmente a través de la web aún que también tiene app's para los SO móviles más importantes, ofrece además de la posibilidad de unificar cuentas de la nube, la posibilidad de unificar también cuentas de redes sociales [3].

Puede indexar toda la información de las nubes en un mismo sitio y catalogar dichos ficheros por categorías.

2.2.2.2 Jolicloud

Este sistema, solo accesible desde la web, empezó como un SO para integrar varios servicios, pero termino centrándose en unificar las cuentas de nube de los usuarios [4].

Ofrece la posibilidad de ejecutar Google Drive de manera que permita editar documentos, pero su mayor problema es que no permite añadir más de una cuenta de una misma nube a no ser que accedamos a sus opciones de pago.

2.2.2.3 Multcloud

Esta plataforma es muy parecida a la anterior, solo que permite la gestión ilimitada de cuentas de una misma nube, además de dar la posibilidad de programar que se realicen copias de ficheros entre las diferentes nubes y así poder crear copias de seguridad [5].

2.2.3 NAS

Por último vamos a hablar del almacenamiento NAS, que es la opción, quizás, más completa. Como ya se dijo vamos a hablar de las versiones comerciales.

Estos servicios ofrecen un sin fin de posibilidades y paquetes complementarios que añaden funcionalidades al sistema, de los cuales podemos destacar la de Cloud Sync. Este paquete es especialmente interesante para nosotros, ya que es el que da posibilidad a que podamos añadir cuentas de la nube a nuestro sistema y sincronizar las mismas. Esto es interesante, pero a priori, el sistema no ofrece una libre navegación y unificada sobre estas cuentas de nube y el sistema. Además no parece que ningún desarrollador permita añadir como servidores de almacenamiento complementarios alguna opción que no sea de sus productos.

Uno de los mayores problemas de estos sistemas es que tiene que ser su producto. Encontramos que se trata de un sistema bastante cerrado, donde aún que se puedan añadir periféricos y algunos terminales al mismo, todos los servicios así como servidores tienen que ser de la propia empresa para que funcionen correctamente, lo que además tiene por contra un sobre coste importante, ya que es un sistema que ofrece unos servicios de buena calidad, pero de un coste bastante elevado.

Uno de los problemas que encontramos también es que no se puede acceder a los servicios a no ser que utilices su producto base. Es decir que si solo quieres usar almacenamiento en nube, no puedes acceder a los sistemas de acceder a las nubes de forma “unificada”, sin que tengas ya el producto comprado, en cambio en nuestro sistema esa posibilidad existe al no exigir al usuario la instalación o adjudicación de algún producto para estas opciones.

2.2.3.1 Synology

Uno de los NAS más importantes es el caso de Synology. Se trata de una empresa que ofrece ese servicio y la cual es bastante extendida. Ofrece un sin fin de funcionalidades, además de la antes mencionada, pero también comparte las mismas carencias [6].

2.2.3.2 Qnap

Es otra empresa del sector. A lo mejor tiene en contra que su software está un poco por detrás del de Synology, pero hay quien opina que su hardware es más competitivo, así que es posible que sea más una cuestión de gusto [7].

2.2.4 Tabla comparativa

En este apartado procederemos a comprar las tecnologías antes descritas con nuestro proyecto. No se compararán las tecnologías en concreto dadas como ejemplos en cada caso, si no el tipo de tecnología que representan, es decir, la nube, unificar la nube y los NAS (comerciales). Para ello se hará una tabla con en la que estarán indicadas dichas tecnologías junto con algunas de las características que se consideran más importantes para este estudio.

Tecnología	Centralización nube	Centralización terminales *	Centralización servidores personales	Almacenamiento activo
La nube	✖	✓ A través de sus aplicaciones de sintonización se puede considerar este caso.	✖	✖
Unificar nube	✓	✖	✖	✖ A pesar de que algunas opciones incorporen la posibilidad de crear una copia de seguridad no es lo suficiente para considerarse como almacenamiento activo, además no es una

				característica propia de estos sistemas.
NAS	✓ Aún que quizás no tenga las suficientes opciones o transparencia necesaria, se considerará que sí centraliza las cuentas de la nube.	✓	✖	✖ Quizás una vez instalados todos los paquetes complementarios que tienen los diferentes sistemas se pueda llegar a considerar una característica, pero no es algo por lo que realmente se caractericen los NAS.
Proyecto	✓	✓	✓	✓

Tabla 1: Comparativa de las posibles soluciones existentes
*** Poder enviarles información y considerarles parte del sistema, no solo para poder acceder al sistema a través de ellos**

Como se puede observar en la tabla (Tabla 1), a pesar de que cada posible solución puede tener sus ventajas o algunas características interesantes propias de cada opción, ninguna de ellas tiene todas características que plantaron el problema inicial del proyecto y por el cual se ha empezado el desarrollo del mismo.

2.3 Tecnologías usadas

En este apartado vamos a hablar de las diferentes tecnologías usadas para el desarrollo del proyecto, así como algunas de las alternativas encontradas y las razones principales por las que finalmente se escogieron.

2.3.1 Almacenamiento (meta-datos)

Vamos a empezar por las diferentes tecnologías para el almacenamiento de la meta-información de nuestro sistema, meta-información como la identificación de los usuarios o la información de los diferentes ficheros que tiene el usuario, como la de su localización, el número de replicas, de acceso, etc. Esta es una parte de la información, pero también se tiene que alojar información sobre los servidores que el usuario quiera añadir a su cuenta, también información sobre las diferentes listas, etc.

2.3.1.1 Servicio de configuración centralizada basado en nodos y almacenamiento clave-valor

La opción elegida para el desarrollo es la de ZooKeeper [8]. Se trata de un sistema de almacenamiento basado en un árbol de nodos, los cuales tienen una funcionalidad de clave-valor, ya que los nodos tienen su nombre y también pueden contener datos. Esto hace de ZooKeeper un sistema con una gran escalabilidad, que es una de las características principales que buscamos de las tecnologías para este fin.

Por otro lado ZooKeeper también soporta alta disponibilidad contando con la capacidad de redundancia de la información, replicándose, lo cual también es de gran interés, puesto que no tengo que preocuparme por replicar la información y mantenerla sincronizada. Por último como característica de gran interés, tenemos que ZooKeeper ofrece una sincronización en las llamadas que se hacen al sistema, tiene un funcionamiento de llamadas distribuidas totalmente implementado y solo necesita de conectarte como cliente al servicio para que puedas realizar todas las acciones ofrecidas sobre el mismo.

Por último además el servicio nos ofrece otras opciones internas, como establecer cuotas o autentificación. Además es un proyecto bastante maduro y con muchos años de desarrollo, el cual continúa hoy en día.

2.3.1.2 Base de Datos no SQL

Aún que estas bases de datos cuenten también con la gran escalabilidad necesaria para el desarrollo del sistema, no incorporan funcionalidades distribuidas de forma nativa, ni las mismas facilidades de uso que pueden ofrecer plataformas como ZooKeeper.

2.3.1.3 Base de Datos SQL

También tenemos la opción de usar algún tipo de distribución de base de datos basado en SQL. Aún que sea un sistema más conocido y “natural” puesto que es una forma de almacenamiento persistente que lleva mucho tiempo en uso y desarrollo y es más intuitivo al poder guardar un registro por fichero, usuario, etc., presenta bastantes desventajas. Para empezar este tipo de almacenamiento es menos escalable que uno pasado en clave-valor o en árbol de nodos, además no suele tener predefinido un soporte de redundancia, que son dos de las principales características que buscamos en nuestro sistema de almacenamiento de tema-información.

2.3.1.4 Tabla comparativa

Vamos a comprar las tecnologías estudiadas en este apartando, indicando las características de mayor interés para mi proyecto, que se verán representados en la siguiente tabla:

Tecnología	Gran escalabilidad	Funcionalidad distribuida	Redundancia	Sincronización de peticiones
Servicio de configuración centralizada basado en nodos y almacenamiento clave-valor	✓	✓	✓	✓
BD no SQL	✓	✘ *	✘	✘
BD SQL	✘	✘	✘ **	✘ *

Tabla 2: Comparativa de tecnologías para el almacenamiento de meta-datos.
*** Puede ofrecer la funcionalidad dependiendo de la distribución.**
**** Puede ofrecer la funcionalidad, pero no de forma trivial.**

Como se puede ver en la tabla (Tabla 2) la tecnología que mejor cubre mis necesidades es ZooKeeper.

2.3.2 Almacenamiento (datos)

Lo que nos interesa de analizar en este apartado es el medio o la tecnología que vamos a usar para almacenar nuestros datos, no los meta-datos como en el caso anterior, si no los datos físicos, es decir los ficheros que queramos manejar. Este es un punto interesante, puesto que en gran parte es sobre lo que gira mi proyecto, para esto tiene que tener la capacidad de manejarse con las nubes más importantes que hay en el mercado y sobretodo la capacidad de permitir que diferentes terminales y servidores puedan almacenar ficheros e interactuar con ellos.

2.3.2.1 Cliente propio

Una de las soluciones más viables es el de la creación de un cliente simple, que no llegue a ser la aplicación que se ofrece como tal, si no un cliente que este a la escucha de ficheros y tenga la gestión de sincronización básica. Esto permitirá poder gestionar ese terminal o servidor de una forma más profunda y personalizada que las otras opciones. De esta forma también se evitan sobrecargas en el cliente con funcionalidades que no serán de utilidad de otras soluciones o la instalación de otros programas terceros, etc.

2.3.2.2 Nubes existentes

Obviamente para el uso de las diferentes nubes y poder trabajar con ellas tendré que hacer uso de las correspondientes api's que ofrecen para su manejo. En este caso estoy

un poco más limitado por lo que ellos ofrecen, pero sin duda tienen la suficiente capacidad para una gestión básica y correcta.

2.3.2.3 Estudio comparativo

Para el caso de los servidores personales existen algunas otras opciones como el FTP, pero es un servicio un poco obsoleto que ofrece muchísimas limitaciones, o la opción de Owncloud, pero además de que ofrecen muchas opciones y más carga en el cliente, no está del todo preparado para el uso que yo le quiero dar.

En el caso de los servidores de la nube, como ya dije, estoy más limitado por el uso de sus api's.

2.3.3 Servidor de aplicaciones

Para la realización de este modulo, se usara un servidor de aplicaciones y para nuestro caso en particular vamos a estudiar dos opciones diferentes. Como se verá más adelante, quizás esta es una de las tecnologías escogidas más por la preferencia que por la necesidad, ya que para el desarrollo del prototipo sirven ambas, pero no ocurre lo mismo para un desarrollo más avanzado.

2.3.3.1 Tomcat

Tomcat [9] es un servidor de aplicaciones desarrollo por Apache y es básicamente integra los servlets y jsp's. Justo por eso es un servidor de aplicaciones fácil de configurar y de usar, además resulta bastante ligero para el sistema que lo hospeda. Por otra parte tiene carencias a la hora de realizar acciones más avanzadas o desarrollos más avanzados, puesto que no integra el JEE entero y para algunas opciones más avanzadas se le tienen que agregar complementos externos al mismo.

2.3.3.2 Glassfish

GlassFish [10] en cambio esta soportado por Oracle y en la práctica se puede decir que es el caso contrario de Tomcat. Es más pesado, pero por otro lado integra totalmente JEE, por lo que tiene más opciones de desarrollo avanzadas.

2.3.3.3 Estudio comparativo

En este caso se ha optado por Tomcat para el desarrollo del prototipo, porque resulta ser un servidor de aplicaciones más ligero y fácil de usar. En cambio a medida que el desarrollo vaya evolucionando es muy probable que empiece a usar GlassFish, puesto que integra opciones de desarrollo más avanzadas de forma nativa.

2.3.4 Lenguaje

Otra tecnología digna de mencionar es el lenguaje de programación usado para el desarrollo del sistema. Obviamente se viene a hablar del lenguaje base y sobre el que se sostiene el sistema, porque para algunos detalles, como por ejemplo la creación de la web, se usan lenguajes complementarios para ciertas funcionalidades.

Analizando un poco las demás tecnologías usadas para el desarrollo y también teniendo en cuenta la potencia de los lenguajes nos encontramos con dos alternativas principales.

2.3.4.1 C

C es un lenguaje muy potente con el que se podría perfectamente desarrollar muchos proyecto de gran alcance, pero se ve limitado en la creación de páginas web o servicios web y como nuestro sistema se sostiene en gran parte sobre un servidor de aplicaciones para el despliegue de la página web o el uso de la interfaz REST, esta limitación es suficientemente importante como para no optar por este lenguaje.

2.3.4.2 Java

Otro lenguaje muy potente y que se adapta perfectamente a ZooKeeper y al resto de nuestras tecnologías es el de java. Además contamos con una plataforma de desarrollo de webs muy madura y de gran alcance como es la de JEE [11], de la que nos podemos hacer uso en el desarrollo tanto de nuestra web, como para la interfaz web, o para el desarrollo de la interfaz REST, la gestión de servidores, etc. Es un lenguaje que además ofrece un gran número de librerías al alcance para este tipo de proyectos.

2.3.4.3 Estudio comparativo comparativa

Básicamente se opta por el lenguaje java por el soporte que este tiene a través de JEE en el desarrollo de las tecnologías web, que es sobre lo que se sustenta mi proyecto.

3 Análisis del problema

En el apartado de diseño se verá un diseño que sería equiparable a la primera versión real del sistema, pero este análisis es sobre lo que consideraría el prototipo inicial, con el que se harán las primeras pruebas de las tecnologías elegidas, así como la determinación de la viabilidad del proyecto y su continuación de desarrollo.

En este análisis se formulan los casos de uso y los requisitos de software para el desarrollo del prototipo. Lo empezaré por los casos de uso y no por los requisitos de usuario, puesto que yo vendría a ser tanto el cliente como el desarrollador, por lo que me parece más natural empezar directamente por los casos de uso, basándome también en los objetivos, en vez de "auto entrevistarme" para conseguir los requisitos de usuario de los que luego se deducirían los casos de uso y los requisitos de software.

3.1 Introducción

Para introducir el análisis y antes de establecer los casos de uso y los requisitos, vamos a hablar del marco regulador y también de los entornos socio-económicos y socio-culturales.

3.1.1 Marco regulador

Vamos a ver tanto el **marco regulador técnico**, como el **marco regulador legal**.

3.1.1.1 Marco regulador técnico

Para el marco regulador técnico se indicarán la metodología utilizada, así como el ciclo de vida.

Metodologías de desarrollo de software

El marco regulador técnico se define en este mismo capitulo, en concreto en los apartados 3.2 y 3.3.

La metodología en la que me he basado en una adaptación propia para realizar el desarrollo de algunos puntos del proyecto, como los casos de uso (3.2) y los requisitos de software (3.3) es el **Métrica v.3** [12]. Es una metodología que sirve para la planificación, desarrollo y mantenimiento de sistemas informáticos, desarrollada y promovida por el Ministerio de Hacienda y Administración Pública del Gobierno de España.

Para consultar el **ciclo de vida** utilizado en el desarrollo se puede consultar 7.2.2.

3.1.1.2 Marco regulador legal

En cuando al **marco regulador legal**, este proyecto se podría ver envuelto principalmente en las siguientes dos leyes:

- **Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal** [13]. Esta ley no se aplica, puesto que nuestro sistema o prototipo no esta pensado, para inicialmente, almacenar ninguna información de carácter personal, tanto para el registro de nuevo usuario como para el uso que el usuario pueda hacer del mismo. Además el sistema no se hace responsable del tipo de datos o ficheros de contenido personal que los usuarios puedan añadir, en caso de que en un futuro se necesitara gestionar este tipo de datos habría que usar los mecanismos de seguridad necesario.

- **Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia** [14]. La ley de Propiedad Intelectual se tendría que tener en cuenta en los casos que el servicio tenga que guardar algún contenido en el sistema con derechos de autor. Si se produce esta circunstancia habría que utilizar un módulo de cifrado (pendiente de futura implementación 8.3.1) que permita ofrecer un mecanismo para el cumplimiento de la misma (o que el usuario almacene el contenido de propiedad cifrado).

3.1.2 Entorno socio-económico

Este sistema se mueve dentro de un entorno que actualmente esta en crecimiento, que es el almacenamiento distribuido. Podemos ver por ejemplo el caso de las nubes o el almacenamiento en NAS, los cuales siguen creciendo.

Mi sistema esta dirigido al mayor número de usuarios, de diferentes edades, que tengan acceso a Internet. Además frente a la nube, mi sistema ofrece una mayor disponibilidad, puesto que si el usuario usa sus propios servidores para almacenar, aún que se cayera el sistema, el usuario seguirá teniendo sus ficheros disponibles. Por otro lado, si por ejemplo, se cae un servidor de los que usa el usuario, tanto propio como de la nube, disfruta de una mayor disponibilidad, al tener la posibilidad de replicar los ficheros más importantes en el resto de servidores.

Por otro lado ofrece una mayor integración de la nube, de la que ofrecen los actuales sistemas, más importantes, de NAS y lo que es más importante, no es un servicio que tienes que comprar adicionalmente, algo que sí ocurre con los mencionados NAS. Además mi sistema, tampoco es un sistema tan cerrado como los NAS, ya que estos solo se manejan con sus propios servidores de almacenamiento, en cambio el mío da la posibilidad de añadir al sistema cualquier servidor propio y guardando la información en tu propio sistema.

Con todo esto el usuario que vaya a usar nuestro sistema puede disfrutar del mayor número de beneficios de los almacenamientos distribuidos más importantes que hay actualmente a la vez que se intentan reducir sus carencias. Con esto el usuario puede disponer de una mayor confianza y de ahorro de tiempo, al tener todo su almacenamiento centralizado.

3.2 Casos de uso (Escenarios)

En este apartado se van a exponer y detallar los diferentes casos de usos que se pueden dar en el sistema, los cuales vienen producidos por un actor externo del sistema, como puede ser un usuario.

3.2.1 Diagrama general de casos de uso

A continuación se mostrará un diagrama general en el cual se podrá ver nuestro único actor externo sobre el sistema, el usuario, con las interacciones que puede realizar:

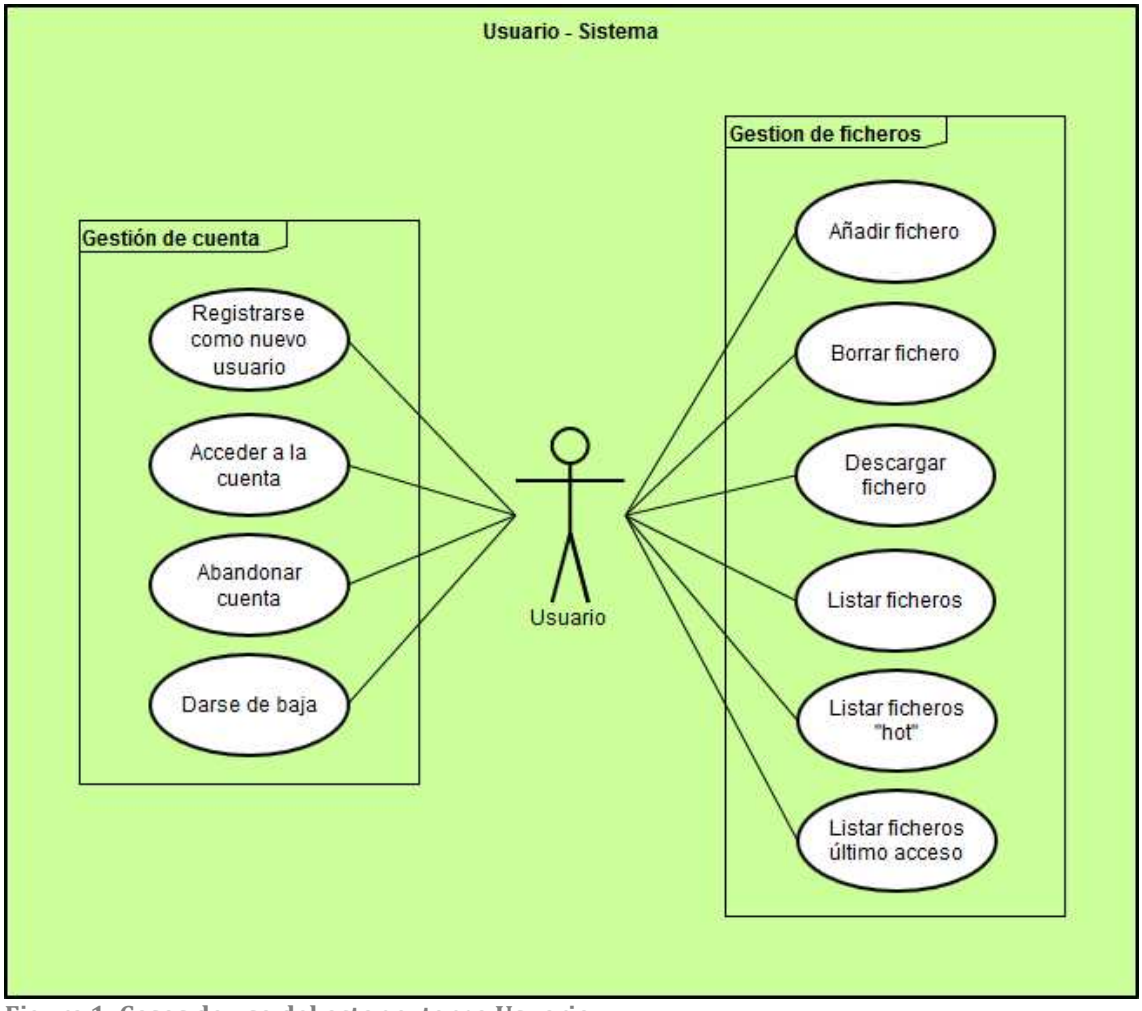


Figura 1: Casos de uso del actor externo Usuario

En este caso no se contempla el actor Administrador, ya que el sistema es autosuficiente. En caso de que se requiera algún tipo de administración, por ejemplo en el caso de ZooKeeper, este se realizará por consola de comandos y de forma interna.

3.2.2 Definición individual de los casos de uso

Además del diagrama general expuesto anteriormente (Figura 1), en el que se pueden ver de forma resumida las acciones que el actor usuario puede realizar sobre el sistema, también detallaremos a continuación estas acciones en tablas con el siguiente formato:

IDENTIFICADOR:	
CASO DE USO	
ACTORES	
OBJETIVO	
ESCENARIO	
PRECONDICIONES	
POST-CONDICIONES	

Tabla 3: Diseño tabla casos de uso

Pasaremos a describir los campos del diseño de tabla de los casos de uso (Tabla 3):

- Identificador: Cada caso de uso tendrá su identificador para facilitar su traza por las fases subsiguientes. El formato que se seguirá es el siguiente: CU-X

- Caso de uso: Es el nombre de la acción que puede realizar uno o varios actores con el fin de conseguir un objetivo determinado.

- Actores: Es una clase de persona, entidad o dispositivo externo que interactúa con el sistema.

- Objetivo: El objetivo que busca conseguir el actor del sistema.

- Escenario: Una descripción escueta del escenario que rodea el caso de uso o los pasos que tiene que seguir el actor.

- Precondiciones: Las condiciones necesarias para que se pueda realizar el caso de uso.

- Post-condiciones: Las situaciones producidas una vez realizado el caso de uso.

IDENTIFICADOR: CU - 1	
CASO DE USO	Registrarse como nuevo usuario
ACTORES	Usuario
OBJETIVO	Registrarse como nuevo usuario en el sistema
ESCENARIO	Rellenar los campos requeridos para el registro y enviar la solicitud
PRECONDICIONES	Tener acceso a la página web
POST-CONDICIONES	En caso de éxito el nuevo usuario se registra en el sistema y se accede directamente a su nueva cuenta

Tabla 4: Caso de uso CU - 1

IDENTIFICADOR: CU - 2	
CASO DE USO	Acceder a la cuenta
ACTORES	Usuario
OBJETIVO	Acceder a la cuenta personal del usuario
ESCENARIO	Rellenar los campos requeridos para el acceso a la cuenta y enviar la solicitud
PRECONDICIONES	Tener acceso a la página web
POST-CONDICIONES	El usuario accede a su cuenta de usuario

Tabla 5: Caso de uso CU - 2

IDENTIFICADOR: CU - 3	
CASO DE USO	Abandonar cuenta
ACTORES	Usuario
OBJETIVO	Abandonar la cuenta de usuario
ESCENARIO	El usuario dispondrá, en todo momento, de un botón para poder abandonar la cuenta
PRECONDICIONES	Tener acceso a la cuenta del usuario y haber accedido en la misma
POST-CONDICIONES	El usuario abandona la su cuenta y es redirigido a la página de login

Tabla 6: Caso de uso CU - 3

IDENTIFICADOR: CU - 4	
CASO DE USO	Darse de baja
ACTORES	Usuario
OBJETIVO	Darse de baja del servicio borrando la cuenta
ESCENARIO	El usuario tiene que acceder a su cuenta particular, donde encontrará la opción de darse de baja.
PRECONDICIONES	Disponer de una cuenta y haber accedido a la misma
POST-CONDICIONES	El usuario borrará su cuenta y contenido, y será redirigido a la página de login

Tabla 7: Caso de uso CU - 4

IDENTIFICADOR: CU - 5	
CASO DE USO	Añadir fichero
ACTORES	Usuario
OBJETIVO	Añadir un fichero en la cuenta del usuario
ESCENARIO	Una vez el usuario ha accedido a su cuenta particular debe seleccionar el fichero que desea subir a su cuenta y enviar la solicitud.
PRECONDICIONES	Acceder a la cuenta del usuario y seleccionar un fichero para subir.
POST-CONDICIONES	El fichero será añadido a la cuenta del usuario y se actualizará la lista con él mismo.

Tabla 8: Caso de uso CU - 5

IDENTIFICADOR: CU - 6	
CASO DE USO	Borrar fichero
ACTORES	Usuario
OBJETIVO	Borrar un fichero de la cuenta del usuario
ESCENARIO	Una vez el usuario ha accedido a su cuenta particular debe seleccionar uno o varios ficheros o todos los ficheros para que sean borrados
PRECONDICIONES	Acceder a la cuenta del usuario y seleccionar los ficheros que se quieran borrar
POST-CONDICIONES	El fichero será eliminado

Tabla 9: Caso de uso CU - 6

IDENTIFICADOR: CU - 7	
CASO DE USO	Descargar fichero
ACTORES	Usuario
OBJETIVO	Descargar un fichero de la lista de ficheros del usuario
ESCENARIO	Cuando el usuario accede a su cuenta se le listarán los ficheros que tienen añadidos a la misma a la vez que dispone de la posibilidad de pulsar sobre el que quiera descargarse.
PRECONDICIONES	Acceder a la cuenta y seleccionar el fichero que se quiera descargar
POST-CONDICIONES	El fichero será enviado al usuario para su descarga

Tabla 10: Caso de uso CU - 7

IDENTIFICADOR: CU - 8	
CASO DE USO	Listar ficheros
ACTORES	Usuario
OBJETIVO	Que el usuario pueda visualizar sus ficheros
ESCENARIO	Cuando el usuario accede a su cuenta dispondrá de la lista con todos los ficheros que tiene añadido en el sistema
PRECONDICIONES	Acceder a la cuenta
POST-CONDICIONES	Lista de los ficheros

Tabla 11: Caso de uso CU - 8

IDENTIFICADOR: CU - 9	
CASO DE USO	Listar ficheros “hot”
ACTORES	Usuario
OBJETIVO	Que el usuario pueda visualizar una lista de los ficheros “hot”
ESCENARIO	Cuando el usuario accede a su cuenta dispondrá de la lista con todos los ficheros que son “hot” en ese momento
PRECONDICIONES	Acceder a la cuenta
POST-CONDICIONES	Lista de los ficheros “hot"

Tabla 12: Caso de uso CU - 9

IDENTIFICADOR: CU - 10	
CASO DE USO	Listar ficheros último acceso
ACTORES	Usuario
OBJETIVO	Que el usuario pueda visualizar una lista de los ficheros manejados en su último acceso
ESCENARIO	Cuando el usuario accede a su cuenta dispondrá de la lista con los ficheros manejados en el último acceso
PRECONDICIONES	Acceder a la cuenta
POST-CONDICIONES	Lista de los ficheros manejados en el último acceso

Tabla 13: Caso de uso CU - 10

3.3 Requisitos de software

MIRAR LAS RELGAS DE AHCER REQUISITOS EN IS

A continuación, se detallarán todas las necesidades y condiciones que debe satisfacer el software describiendo los requisitos de software del análisis. Estos requisitos ayudan a complacer los casos de uso, y además los objetivos y necesidades básicas que considero necesarias. Estos requisitos son sobre los que se va a cimentar el resto del proyecto, siendo una referencia para verificar el prototipo y son lo mínimo e indispensable que se debe cumplir.

Se hace distinción entre dos tipos de requisitos:

- Funcionales
- No funcionales

La estructura de los requisitos es la siguiente:

IDENTIFICADOR:	
DEFINICIÓN:	
CREADOR:	
PRIORIDAD: Alta / Media / Baja	**NECESIDAD:** Obligatorio / Deseable / Opcional
ESTABILIDAD:	
DESCRIPCIÓN:	

Tabla 14: Formato tabla de requisitos

Pasaremos a describir los campos de formado de tabla de los requisitos (Tabla 14):

- Identificador: Cada requisito software tendrá su identificador para facilitar su traza por las fases subsiguientes. El formato que se seguirá es el siguiente:
  - Funcionales: su identificador será RF-X.
  - No funcionales: su identificador será RNF-X.
- Definición: Definición resumida del requisito en cuestión.
- Creador: La persona o entidad creadora del requisito.
- Prioridad: Las medidas de prioridad serán alta, media o baja para que el desarrollador pueda decidir la planificación de la producción según las mismas.

- Necesidad: Los requisitos de software obligatorios se marcarán como tales. Estos no son negociables. El resto pueden estar sujetos a negociación o cambios.

- Estabilidad: Algunos requisitos se pueden ser fijos sobre la vida esperada del software, mientras que otros pueden depender de las decisiones de diseño o implementación que se tomen durante el desarrollo.

- Descripción: Descripción más detallada del requisito.

3.3.1 Funcionales

Especifican qué debe hacer el producto, concretan el propósito del mismo. Se derivan de los requisitos de usuario y casos de uso planteados en el apartado anterior (3.2).

IDENTIFICADOR: **RF - 1**	
DEFINICIÓN: Que los usuarios puedan registrarse en el servicio	
CREADOR: Teodor Sergeev Totev	
PRIORIDAD: Alta	**NECESIDAD:** Obligatorio
ESTABILIDAD: Durante toda la vida del sistema	
DESCRIPCIÓN:	Permitir a los usuarios registrarse en el servicio, siempre y cuando no intenten utilizar un nombre de usuario ya existente, en este caso se mostrará el error correspondiente.

Tabla 15: Requisito funcional RF - 1

IDENTIFICADOR: **RF - 2**	
DEFINICIÓN: Que los usuarios puedan acceder a su cuenta	
CREADOR: Teodor Sergeev Totev	
PRIORIDAD: Alta	**NECESIDAD:** Obligatorio
ESTABILIDAD: Durante toda la vida del sistema	
DESCRIPCIÓN:	Permitir al usuario que pueda acceder a su cuenta particular, siempre y cuando introduzca un usuario y contraseña correctos, en este caso se mostrará el error correspondiente.

Tabla 16: Requisito funcional RF - 2

IDENTIFICADOR: **RF - 3**
DEFINICIÓN: Que los usuarios puedan abandonar su cuenta

IDENTIFICADOR: **RF - 3**	
CREADOR: Teodor Sergeev Totev	
PRIORIDAD: Alta	**NECESIDAD:** Obligatorio
ESTABILIDAD: Durante toda la vida del sistema	
DESCRIPCIÓN:	Permitir al usuario la capacidad de cerrar y abandonar su cuenta.

Tabla 17: Requisito funcional RF - 3

IDENTIFICADOR: **RF - 4**	
DEFINICIÓN: Que los usuarios puedan darse de baja en el servicio	
CREADOR: Teodor Sergeev Totev	
PRIORIDAD: Alta	**NECESIDAD:** Obligatorio
ESTABILIDAD: Durante toda la vida del sistema	
DESCRIPCIÓN:	Permitir a los usuarios darse de baja del servicio, borrando todo su contenido.

Tabla 18: Requisito funcional RF - 4

IDENTIFICADOR: **RF - 5**	
DEFINICIÓN: Permitir a los usuarios poder añadir un fichero	
CREADOR: Teodor Sergeev Totev	
PRIORIDAD: Alta	**NECESIDAD:** Obligatorio
ESTABILIDAD: Durante toda la vida del sistema	
DESCRIPCIÓN:	Que los usuarios puedan añadir un fichero nuevo a su cuenta, siempre y cuando no tenga el nombre de uno ya existente, en este caso se mostrará el error correspondiente.

Tabla 19: Requisito funcional RF - 5

IDENTIFICADOR: **RF - 6**	
DEFINICIÓN: Permitir a los usuarios poder borrar un fichero	
CREADOR: Teodor Sergeev Totev	
PRIORIDAD: Alta	**NECESIDAD:** Obligatorio
ESTABILIDAD: Durante toda la vida del sistema	

IDENTIFICADOR: RF - 6	
DESCRIPCIÓN:	Que los usuarios puedan borrar un fichero de su cuenta. Además se le permitirá que pueda seleccionar más de un fichero, para borrarlos a la vez o poder seleccionar que se borren todos los ficheros de su cuenta.

Tabla 20 Requisito funcional RF - 6

IDENTIFICADOR: RF - 7	
DEFINICIÓN: Permitir a los usuarios poder recuperar un fichero	
CREADOR: Teodor Sergeev Totev	
PRIORIDAD: Alta	**NECESIDAD:** Obligatorio
ESTABILIDAD: Durante toda la vida del sistema	
DESCRIPCIÓN:	Que los usuarios puedan descargar un fichero de su cuenta al terminal desde el cual están conectados en ese momento

Tabla 21: Requisito funcional RF - 7

IDENTIFICADOR: RF - 8	
DEFINICIÓN: Que los usuarios puedan ver los ficheros que tienen en su cuenta	
CREADOR: Teodor Sergeev Totev	
PRIORIDAD: Alta	**NECESIDAD:** Obligatorio
ESTABILIDAD: Durante toda la vida del sistema	
DESCRIPCIÓN:	Se le presentará en una lista todos los ficheros de los que dispone el usuario en ese momento en su cuenta. Esta lista se presenta al usuario nada más acceder a su cuenta y además se actualizará cada vez que el usuario añada un nuevo fichero o borre alguno, algunos o todos sus ficheros. Además también se mostrará una lista de los ficheros “hot, como también de los ficheros tratados en el último acceso a la cuenta.

Tabla 22: Requisito funcional RF - 8

IDENTIFICADOR: RF - 9
DEFINICIÓN: Tamaños máximo de ficheros

IDENTIFICADOR: **RF - 9**	
CREADOR: Teodor Sergeev Totev	
PRIORIDAD: Alta	**NECESIDAD:** Obligatorio
ESTABILIDAD: Durante toda la vida del sistema	
DESCRIPCIÓN:	El tamaño máximo permitido para los ficheros que el usuario quiera subir será de 500KB, en caso contrario se mostrará el error correspondiente

Tabla 23: Requisito funcional RF - 9

IDENTIFICADOR: **RF - 10**	
DEFINICIÓN: Definición de formatos	
CREADOR: Teodor Sergeev Totev	
PRIORIDAD: Alta	**NECESIDAD:** Obligatorio
ESTABILIDAD: Durante toda la vida del sistema	
DESCRIPCIÓN:	El nombre de usuario tiene que contener solo caracteres alfanuméricos. Al igual que para los ficheros que el usuario pueda subir al sistema, con la diferencia que en este caso también se aceptará el carácter “.”. En caso contrario se mostrará el error correspondiente

Tabla 24: Requisito funcional RF - 10

IDENTIFICADOR: **RF - 11**	
DEFINICIÓN: Acceso a través de una página web	
CREADOR: Teodor Sergeev Totev	
PRIORIDAD: Alta	**NECESIDAD:** Obligatorio
ESTABILIDAD: Durante toda la vida del sistema	
DESCRIPCIÓN:	El usuario tendrá que acceder al sistema a través de una interfaz de usuario web, para lo cual tiene que disponer de acceso a la red.

Tabla 25: Requisito funcional RF - 11

3.3.2 No funcionales

Describen como debe funcionar el sistema y se refieren a los requisitos que no especifican alguna información para guardar o alguna función para realizar.

IDENTIFICADOR: RNF - 1	
DEFINICIÓN: Que el sistema este escrito en el lenguaje java y JEE.	
CREADOR: Teodor Sergeev Totev	
PRIORIDAD: Alta	**NECESIDAD:** Obligatorio
ESTABILIDAD: Durante toda la vida del sistema	
DESCRIPCIÓN:	El lenguaje base para el desarrollo del proyecto tiene que ser java, para poder soportar aspectos de suma importancia para el sistema, como la plataforma JEE

Tabla 26: Requisito no funcional RNF - 1

IDENTIFICADOR: RNF - 2	
DEFINICIÓN: Usar Apache ZooKeeper para centralizar y sincronizar el sistema	
CREADOR: Teodor Sergeev Totev	
PRIORIDAD: Alta	**NECESIDAD:** Obligatorio
ESTABILIDAD: Durante toda la vida del sistema	
DESCRIPCIÓN:	Que el sistema usado para sincronizar la información y centralizarla sea ZooKeeper

Tabla 27: Requisito no funcional RNF - 2

IDENTIFICADOR: RNF - 3	
DEFINICIÓN: Que el servidor de aplicaciones sea Apache Tomcat.	
CREADOR: Teodor Sergeev Totev	
PRIORIDAD: Alta	**NECESIDAD:** Obligatorio
ESTABILIDAD: Durante toda la vida del sistema	
DESCRIPCIÓN:	El servidor de aplicaciones usada para el desarrollo y despliegue del sistema debe ser Tomcat

Tabla 28: Requisito no funcional RNF - 3

IDENTIFICADOR: RNF - 4	
DEFINICIÓN: Que este en castellano.	
CREADOR: Teodor Sergeev Totev	
PRIORIDAD: Alta	**NECESIDAD:** Obligatorio

IDENTIFICADOR: **RNF - 4**	
ESTABILIDAD: Durante toda la vida del sistema	
DESCRIPCIÓN:	El prototipo debe ser en castellano

Tabla 29: Requisito no funcional RNF - 4

IDENTIFICADOR: **RNF - 5**	
DEFINICIÓN: Los mensajes de error tienen que ser claros	
CREADOR: Teodor Sergeev Totev	
PRIORIDAD: Alta	**NECESIDAD:** Obligatorio
ESTABILIDAD: Durante toda la vida del sistema	
DESCRIPCIÓN:	Los mensajes de error que se produzcan deben de ser claros y que ayuden a guiar al usuario

Tabla 30: Requisito no funcional RNF - 5

IDENTIFICADOR: **RNF - 6**	
DEFINICIÓN: Ley de propiedad intelectual	
CREADOR: Teodor Sergeev Totev	
PRIORIDAD: Alta	**NECESIDAD:** Obligatorio
ESTABILIDAD: Durante toda la vida del sistema	
DESCRIPCIÓN:	El prototipo, como el sistema, en caso de que así lo requiera la situación, deben cumplir con lo establecido en la ley: **Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia** [14]

Tabla 31: Requisito no funcional RNF - 6

3.4 Matriz de trazabilidad

Se procede a definir la matriz de trazabilidad entre los requisitos de software y los casos de uso para comprobar que todos los casos de uso vienen integrados en algún requisito. Esta matriz tiene que justificar que los requisitos cubren a todos los casos de uso presentados.

	CU - 1	CU - 2	CU - 3	CU - 4	CU - 5	CU - 6	CU - 7	CU - 8	CU - 9	CU - 10
RF - 1	✓									
RF - 2		✓								
RF - 3			✓							
RF - 4				✓						
RF - 5					✓					
RF - 6						✓				
RF - 7							✓			
RF - 8								✓	✓	✓
RF - 9					✓					
RF - 10	✓	✓			✓					
RF - 11	✓	✓								
RNF - 1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
RNF - 2	✓	✓		✓	✓	✓	✓	✓	✓	✓
RNF - 3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
RNF - 4	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
RNF - 5	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
RNF - 6					✓					

Tabla 32: Matriz de trazabilidad de requisitos de software frente a casos de uso

Como se puede apreciar en la matriz de trazabilidad (Tabla 32) todos los requisitos cubren al menos un caso de uso y todos los casos de uso están soportados por al menos

un requisito. Además se pude observar que, en el caso de los requisitos no funcionales y al ser de cómo se debería hacer o implantar algo, están presenten en casi todos los casos de uso.

4 Diseño de la solución técnica

4.1 Introducción

Ene este capitulo vamos a ver el diseño que se ha realizado para el proyecto, así como algunas de las diferentes alternativas propuestas en diferentes partes del desarrollo.

4.2 Diseño general

Para poder realizar un sistema centralizado, que se dedique a conectar todos los servidores deseados al mismo, controlar su acceso y a la vez también controlar el acceso de los usuarios, así como sus peticiones, el sistema tiene que contar de varias partes o módulos y hacer que las mismas funcionen de una forma sincronizada entre ellas.

En la siguiente figura podemos ver los principales módulos del diseño inicial del sistema con las respectivas interacciones entre las mismas.

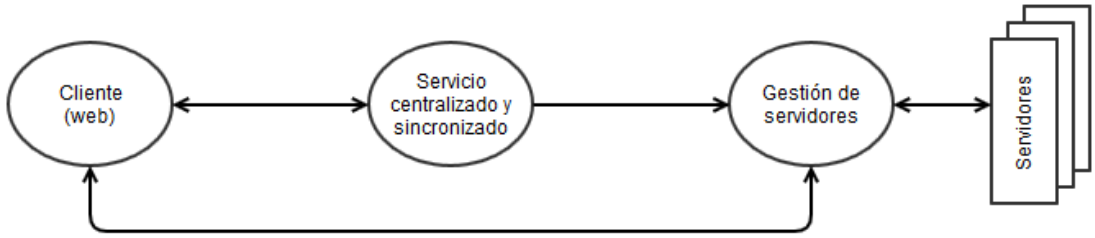


Figura 2: Diseño inicial del sistema

Como se puede ver en la figura (Figura 2) existen 3 módulos principales:

- Cliente (web): Es el módulo encargado de presentar la interfaz web al usuario y también gestionar algunas funcionalidades por la parte del servidor. Este módulo esta localizado en el servidor de aplicaciones, en este caso Apache Tomcat (2.3.3.1), al menos para el prototipo y las primeras versiones.

- Servicio centralizado y sincronizado: Este módulo es el que se encarga de gestionar, manejar y guardar toda la meta-información del sistema, crear una alta disponibilidad a la misma, así como sincronizar peticiones a la misma y esta gestionado por Apache ZooKeeper (2.3.1.1).

- Gestión de servidores: Este es el módulo que se encarga de realizar las comunicaciones con los diferentes servidores, enviar o recibir ficheros de los mismos, entre otros. Este módulo también esta localizado en el servidor de aplicaciones, también por Apache Tomcat, al menos para el prototipo y las primeras versiones.

También se puede apreciar que todos los módulos se relacionan entre sí y además casi en todos los casos estas relaciones son bidireccionales. Estas relaciones entre los módulos se irán detallando, para cada caso particular de funcionalidades, en este diseño.

4.2.1 Módulo “Servicio centralizado y sincronizado”

La parte central del proyecto tiene que ser el sistema que se dedique a realizar las comunicaciones entre los usuarios y los servidores, no de forma directa, ya que de esto se encarga el módulo de gestión de servidores, pero este módulo es el que sabe donde se encuentra toda la información y lo que hay que recuperar, guardar o borrar.

Este modulo del sistema tiene que ser lo suficientemente robusto para que pueda soportar una escalabilidad importante, en caso de que haya muchos usuarios que opten por agregar sus propios servidores al sistema y que a su vez tengan una gran cantidad de información. Además el sistema tiene que ser capaz de ofrecer una alta disponibilidad y recibir varias peticiones a la vez y sincronizar correctamente las mismas, sobretodo cuando se trate de acceder al mismo servidor y lo que ya sería todavía más delicado, al mismo fichero. Para resolver estos problemas que se nos plantean para el módulo podemos decir que tenemos dos opciones:

- Diseñar un servicio que nos ofrezca todas estas prestaciones, además de algunas otras funcionalidades.

- Usar alguna herramienta o sistema que ya nos ofrezca algunas o todas nuestras necesidades para poder cumplir con las exigencias del modulo.

En nuestro caso, como ya se ha indicado con anterioridad, vamos a optar por la segunda opción y después de estudiar algunas de las diferentes opciones que existen y a las que podríamos haber optado, elegimos ZooKeeper.

4.2.2 Módulo “Cliente (web)”

Una vez resuelto él que se podría considerar como el módulo principal o central de nuestro sistema, tenemos que ofrecer de alguna forma el acceso a los clientes o usuarios, que se vayan a registrar y usar nuestro sistema.

Para ello vamos a tener otro módulo, que es el que ofrecerá el acceso a los clientes al sistema. Para proporcionar este acceso tenemos las siguientes formas de hacerlo:

- Primero tenemos la opción, que podríamos definir como, “por defecto” que sería una interfaz web, desde la cual el usuario puede registrarse, acceder al sistema y gestionar su cuenta. Es importante que esta web este diseñada de forma amena, limpia y que ofrezca una vista correcta en todos los tipos de navegadores y dispositivos desde los que se pueda acceder a la misma.

- Otra opción sería a través de una aplicación, que además podría ofrecer al usuario otro tipo de ventajas que no ofrecería el acceso desde la web.

  Algunas de estas ventajas son, por ejemplo, poder trabajar de forma offline guardando ficheros en el sistema y que una vez se tenga acceso a la red esté se sincronice. Además también podría ofrece la opción de acceso a la información o ficheros más usados por el usuario o los ficheros "hot" y/o de los ficheros del último acceso online, es decir, se haría una copia local de las listas de ficheros "hot" y/o de último acceso para que en caso de quedarse el usuario sin acceso a la red, poder acceder a este tipo de fichero. Esto pasaría con los ficheros de un tamaño máximo determinado, incluso se podría dejar que el usuario pueda definir lo que quiera que se pueda acceder o lo que no de forma offline, sin límites de espacio o con límite, unos tipos de ficheros u otros, etc.

  Todos estos mecanismos ofrecerán al usuario la posibilidad de continuar algún trabajo que este realizando, en un momento dado, a pesar de quedarse sin conexión.

  Este tipo de aplicaciones se ofrecerán de forma oficial, pero quedan fuera de la planificación de la primera versión del sistema y se realizarán como trabajos futuros (8.3.2.1).

- Por último tendríamos una última opción que sería ofrecer al usuario una interfaz basada en REST y que nuestro sistema se pueda usar como un servicio web, para que el mismo usuario pueda crear sus propias formas de acceso a nuestro servicio y pueda aprovecharse de las máximas ventajas posibles. Además nuestras aplicaciones y página web, también podrían hacer uso de esta interfaz, pero como en el caso anterior de las aplicaciones, esta opción queda fuera de la primera versión del servicio y se realizará en futuras versiones (8.3.2.1).

Como ya se ha indicado en el diseño general, para este módulo se usará el servidor de aplicaciones Apache Tomcat, sobre este servidor se creará la web dinámica que se ofrecerá como opción más común o "por defecto" de nuestro servicio.

Una vez realizado el modulo encargado de ofrecer una vista al usuario, hay que hacer que este se comunique de forma correcta con el modulo encargado de sincronizar todo y manejado por ZooKeeper, de esta forma un usuario puede pedir un fichero sin tener la necesidad de saber donde esta alojado y el modulo central se encarga de buscarlo, solicitarlo al servidor pertinente y ofrecérselo al usuario que ha hecho la petición.

4.2.3 Modulo "Gestión servidores"

Una vez realizado todo lo necesario para que el usuario se pueda comunicar con el sistema y también tener un modulo encargado de gestionar la meta-información, tenemos que crear todas las comunicaciones necesarias para que al sistema se puedan añadir, eliminar o gestionar servidores y comunicarse con los mismos, así como

terminales, servidores en la nube o tener un servidor personal para encolar peticiones (temporales). Este módulo queda excluido del primer prototipo y quedaría como una implementación para futuros prototipos (8.3.1).

También, como trabajos fututos, se podrían incluir algunas otras funcionalidades más avanzadas en la gestión de los servidores, como por ejemplo la posibilidad de autentificación (8.3.2.2).

4.3 Propuestas de diseño

Para el funcionamiento global del producto final, o lo que vendría a ser la primera versión estable, se presentaran dos propuestas diferentes, en cuanto a la forma de uso del mismo, que tiene una diferencia destacada.

- Una opción es totalmente personal, ya que es el usuario el que se encargaría de todo, es decir, instalar la web con el servidor en algún host, instalar un servidor para ZooKeeper y obviamente encargarse de ir añadiendo los servidores que quiera.

- La segunda opción y en principio la opción más viable, simple y más cómoda, contaría con una web y/o aplicación facilitada por la empresa desarrolladora, así como un servidor donde tener instalado ZooKeeper, de forma que el usuario no se tenga que preocupar de nada más que te registrarse y usar el servicio.

4.3.1 Propuesta 1

Esta opción tiene el atractivo principal de que todo se gestiona de forma personal y que el usuario tiene el control absoluto del sistema, pero presenta dificultades como un alto conocimiento de gestión de sistemas por parte del usuario, cosa que para un usuario convencional puede suponer un gran problema, además de suponer dificultades a la hora de actualizar funcionalidades del sistema por parte del desarrollador o de la misma web. Por último además el sistema carecería de ninguna funcionalidad de la nube (como por ejemplo tener un espacio como servidor temporal), ya que no dispondría del servidor de la empresa desarrolladora. Se le podría dar acceso a alguno de estos servicio, por defecto, y tenerlos añadidos a ZooKeeper como servidores alternativos para su uso, pero rompería con uno de los atractivos más importantes de esta alternativa, que es el control absoluto sobre el sistema.

Podríamos decir que este diseño es más viable para una empresa, por el conocimiento que requiere y por el hecho de tener el control sobre el sistema, pero para otro tipo de usuarios, sobretodo más ocasionales, puede ser algo complejo y molesto. En este caso también tendrían la ventaja de poder gestionar todos los usuarios que acceden al sistema y poder definir con facilidad los accesos a los servidores, en caso de tratarse de una gestión empresarial. Al final se resume como la ventaja de tener el control absoluto sobre el sistema.

4.3.2 Propuesta 2

Esta segunda opción tiene como principal atractivo el acceso fácil y rápido al sistema por todo tipo de usuarios, además de ofrecer una mayor facilidad de uso. Esta propuesta no requiere que el usuario tenga que configurar nada más que los servidores que quiera añadir al sistema para su uso. Además también ofrece al usuario la posibilidad de despreocuparse de gestionar la meta-información del sistema, encargarse de gestionar redundancias sobre ZooKeeper, etc. cosas que a lo mejor en el caso anteriores se podría llegar a necesitar. También se despreocuparía de tener que actualizar el sistema con las nuevas funcionalidades y opciones que se vayan añadiendo, ya que es el desarrollador el que se encarga de realizar todas estas tareas, a excepción de que haya que realizar algún cambio en los clientes instalados sobre los servidores añadidos.

4.3.3 Tabla comparativa

Propuestas	Control absoluto sobre el sistema (usuario)	Facilidad de uso (usuario)	Facilidad de actualización (administración y usuario)	Destinado a mayor tipo de usuarios
Propuesta 1	✓	✗	✗	✗
Propuesta 2	✗	✓	✓	✓

Tabla 33: Comparativa de propuestas de diseño

La propuesta elegida finalmente será la **propuesta 2**, ya no solo por las ventajas que ofrece (Tabla 33), tanto a los usuarios como a mi mismo, si no que la ventaja que ofrece la propuesta 1 no la considero de tanto peso y sobretodo esta dirigida para un tipo de usuarios muy minoritarios a los que probablemente se les podría ofrecer alguna gestión complementaría en un futuro, en caso de necesidad. En cambio la propuesta 2 ofrece más ventajas para un mayor tipo de usuarios, una de las cosas que se pretende con el sistema, llegar al mayor número y tipo de usuario.

4.4 Diseño de la propuesta elegida

Hay que decir que en este apartado se hablará de cosas que valen perfectamente para ambas propuestas (puesto que las diferencias son principalmente a nivel de gestión y no de diseño), como la gestión de la meta información o las primeras funcionalidades del sistema, tengo que decir que de haber algunos pequeños matices de diseño entre uno y otro se considerarían los de la propuesta elegida.

Para ellos pasaremos a explicar el diseño más detallado de cada uno de los 3 módulos principales de nuestro sistema, de los que ya se hablo con anterioridad, que serían el módulo Cliente (web), Servicio centralizado y sincronizado y gestión de servidores.

La clasificación de los módulos, anteriormente mostrada (Figura 2), es una clasificación de nivel alto. Dado que el módulo de Gestión de servidores esta contenido en el código de la web, que es gestionado por el servidor de aplicaciones, junto con el módulo de Cliente (web) y las funcionalidades sobre el sistema en sí, se puede considerar como un conjunto entero, mientras que el Servidor centralizado y sincronizado estará gestionado por el servidor de ZooKeeper. Para poder ver esto mejor, se puede ver en la siguiente figura como sería el sistema de una forma más detallada.

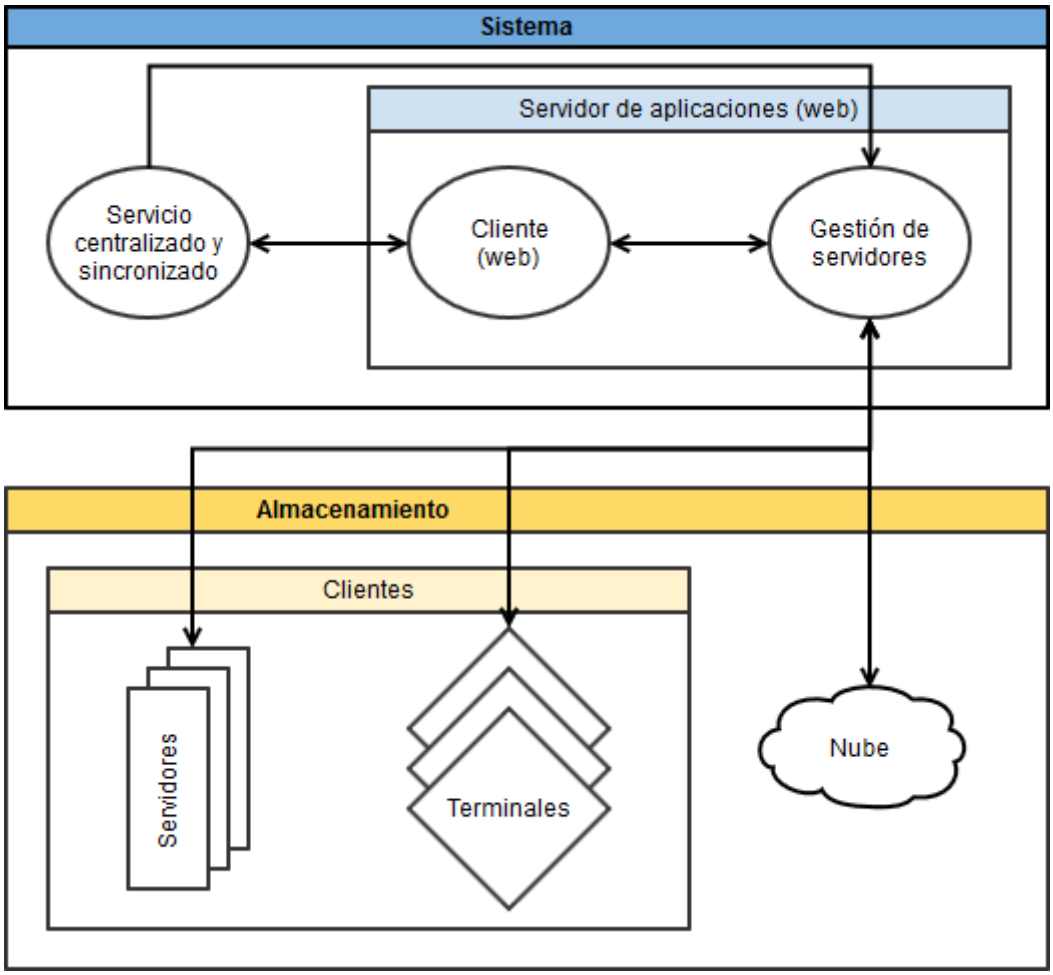


Figura 3: Diagrama de diseño detallado

En este diagrama (Figura 3) se puede ver de una forma más clara la delimitación entre el sistema y los almacenamientos que el mismo tiene que centralizar.

En el primer caso, el sistema, se puede ver que los dos módulos Cliente (web) y Gestión de servidores, están en el mismo contenedor y están gestionado por el mismo servidor de aplicaciones. En cambio el módulo se Servicio centralizado y sincronizado queda fuera de este contenedor porque está gestionado por otro servicio, que en este caso es ZooKeeper el, que también da soporte al mismo en otro servidor.

Por otro lado está el almacenamiento, que en este caso esta compuesto por servidores personales (entre los que se puede incluir algún servidor temporal del usuario), terminales (los cuales pueden ser móviles, PC's, etc) y la nube (cuentas de la nube).

También se puede ver como los servidores y terminales estarían agrupados por la gestión de los clientes en cambio la nube no lo tiene, ya que para su acceso se utiliza la api proporcionado por el desarrollador y las funciones de las que dispone.

Ahora pasaremos a detallar los diseños de las dos partes de las que se compone el sistema, el servidor de aplicaciones y el servidor de ZooKeeper.

4.4.1 Servidor de aplicaciones

En este apartado vamos a hablar tanto del módulo Cliente (web) como el de Gestión de servidores. Hasta ahora se han definido como módulo diferentes, de forma conceptual, por la importancia que tienen, pero realmente ambas características están definidas en el mismo conjunto y todo esta gestionado en el lugar donde esta la web con el servidor de aplicaciones.

Cliente (web)

Como ya se ha dicho anteriormente, el desarrollo de la web se realizará con la plataforma JEE y tiene varias partes claramente definidas.

Una de estas partes sería la encargada de ofrecer al usuario la interfaz de usuario mediante la cual el mismo se podrá comunicar con el sistema y gestionar su cuenta.
Esta interfaz de usuario tendrá que ofrecer al mismo una página inicial y de bienvenida que ofrecerá al mismo la posibilidad de entrar en su cuenta personal o de poder registrarse para tener esa cuenta.

Una vez dentro de la cuenta se tendría que ofrecer al usuario la posibilidad de poder gestionar los ficheros que tenga dentro de la misma. Para poder gestionar sus ficheros, se ofrecerá una lista de los mismos nada más entrar a la cuenta, desde la cual se podrán descargar los mismos como también borrar. Para subir un fichero tiene que existir un formulario para ese fin.

Además de la gestión de los ficheros, también se tiene que ofrecer la posibilidad de crear carpetas, las cuales también se tienen que poder crear, borrar, etc.

Los ficheros y las carpetas también tienen que ofrecer la posibilidad de ser renombrados, así como de ser movidos o copiados, no replicados en los diferentes servidores, si no copiado en el mismo o en otra ruta. También tienen que ofrecer la posibilidad de ser replicados en los diferentes servidores, en todos o en algunos elegidos entre la lista de los servidores que estén agregados a la cuenta del usuario.

A parte de la lista principal de los ficheros del usuario, este puede tener la posibilidad de visualizar una lista de los ficheros “hot” o los accedidos en la anterior sesión. Los ficheros “hot” se determinan según el número de accesos a los mismos de acuerdo a unos criterios basados en el número de acceso en un marco de tiempo. Esta lista se actualizará cada vez que un fichero pasa a ser “hot” o cuando se acceda a la cuenta. La

lista de los ficheros accedidos en la última sesión se actualiza al salir de la sesión actual. Además ambas listas se actualizarán cuando se borre algún fichero que este en ellas.

La interfaz también tiene que ofrecer al usuario poder gestionar algunos aspectos de la cuenta, como por ejemplo poder cambiar la contraseña, entre otras, como poder salir de la misma o darse de baja, borrando la misma, así como todo su contenido.

Gestión de servidores

También tiene que ofrecer al usuario la posibilidad de añadir y gestionar los servidores que quiera ir añadiendo el usuario a su cuenta.

Para ello tiene que tener la posibilidad de añadir o borrar servidores que además en caso de ser servidores propios, poder asignar a los mismos una contraseña o autentificación o por ejemplo una posibilidad de cifrar la información, en cuyo caso, teniendo en cuenta que solo se podrá acceder a la información mediante nuestro sistema. Además se dejará la posibilidad al usuario de definir algunos servidores como servidor de “pendiente”. Este servidores se usará cuando algún servidor de los que el usuario quiera guardar algo no responda, de forma que cuando el servidor vuelva a estar activo, el sistema se encarga de mover el fichero desde el servidor “pendientes”.

Para los terminales, como PC’s o tablets, que en términos prácticos serán tratados como los servidores personales, pero en este caso se ofrecerá dentro se los mismos una carpeta donde se manejará toda la información. Para los terminales también se ofrecerá la posibilidad de enviar un fichero a alguno, sin la necesidad de que se tenga un registro en el sistema.

Las replicas se contarán solo si se encuentra el mismo fichero en varios servidores, pero si la replica del fichero esta en el mismo servidor no contarlo como una replica real o un respaldo de dicho servidor.

Para la gestión de estos servidores y terminales se usará un cliente creado para este fin, un cliente simple que simplemente estará escuchando las peticiones. Para la gestión y acceso de los servidores en nube se usarán las correspondientes api’s y además tendrá que realizarse la autenticación pertinente.

En caso de que un fichero se modifique o borre directamente accediendo a alguno de los servidores, una de las posibilidades que le queda al sistema es que cuando el usuario se autentifique, el sistema haga una actualización general de sus datos, comprobando que todo este en su sitio, pero esto puede tener mucha carga si se contiene mucha información. La otra opción en este caso es que cuando el usuario quisiera acceder a un fichero y el sistema detecta que ese fichero ha sido modificado o borrado, haga una actualización de todo el servidor donde se encuentra dicho fichero.

Otros módulos

También hay que encargarse de la parte de la web que tiene que comunicarse con el módulo encargado de centralizar la información, que es el gestionado por ZooKeeper. Para realizar este apartado, se utiliza el api de ZooKeeper.

4.4.2 Módulo “Servicio centralizado y sincronizado”

Este módulo se puede considerar como el módulo central del sistema y de gran importancia, como ya se ha indicado con anterioridad, ya que es el encargado de sincronizar y guardar la meta-información del sistema.

Pasaremos a hablar sobre la estructura interna de la información que maneja ZooKeeper, de cómo se organiza la jerarquía de los nodos, los usuarios con sus ficheros y servidores, etc. además de las estructuras o tablas de información que tiene cada fichero o servidor añadido a la cuenta del usuario.

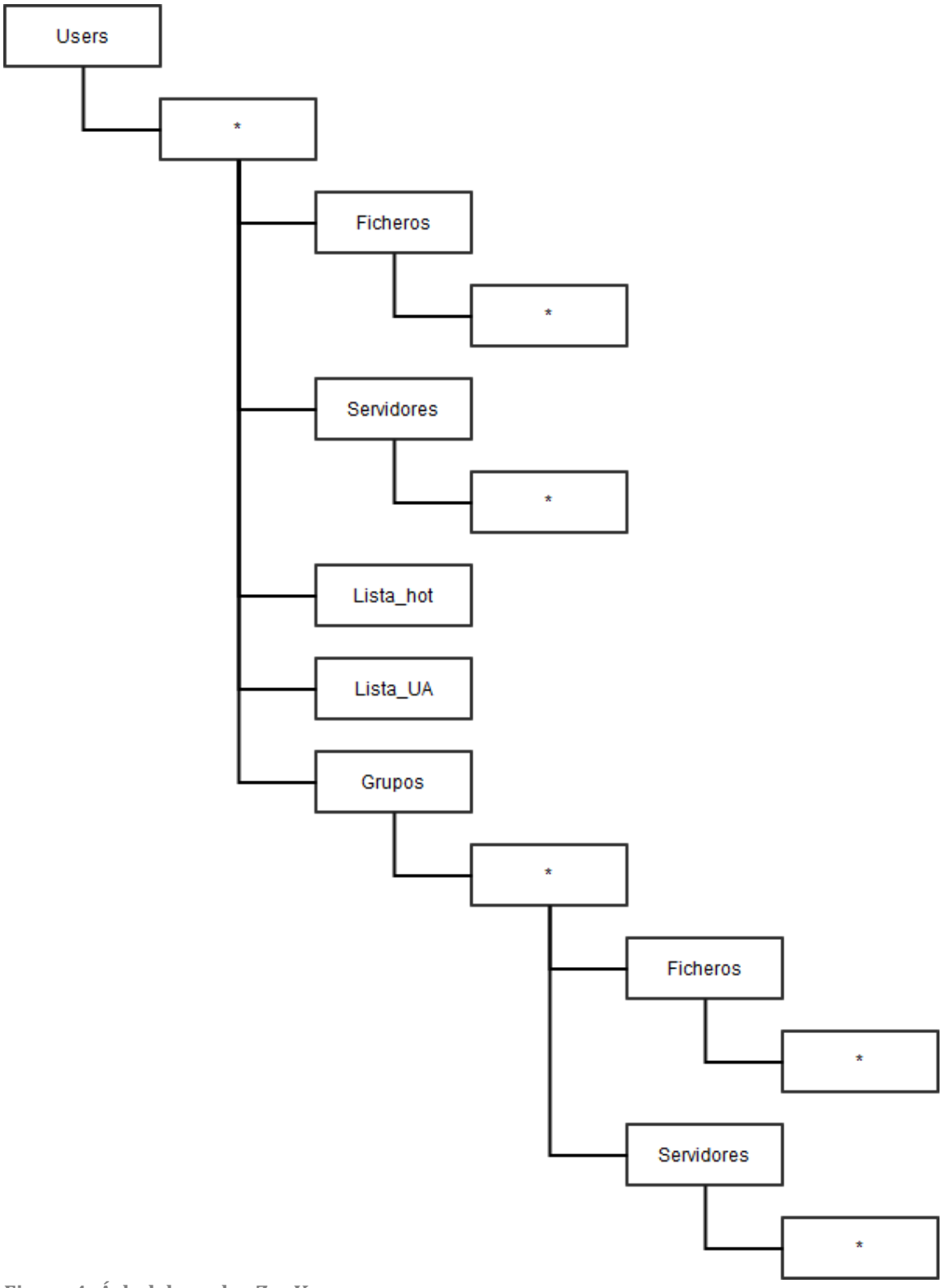


Figura 4: Árbol de nodos ZooKeeper

El esquema de nodos representado en la figura (Figura 4) puede variar un poco en un diseño final, pero la idea básica sería la que se muestra.

Como podemos ver hay un nodo principal que es Users y de los que cuelgan los diferentes usuarios. En el nodo del nombre del usuario, además se puede guardar

información acerca del usuario, en caso de necesidad. Una vez creado el nodo del usuario, del mismo cuelgan los nodos Ficheros, Servidores, Lista_hot, Lista_UA, Grupos. Los nodos de Ficheros y Servidores a su vez contienen nodos con los nombres de cada fichero o servidor que tiene el usuario y cada nodo de estos tiene una tabla que contiene la meta-información del mismo.

Para los ficheros una posible tabla sería la siguiente:

Dirección	
Número de accesos	
Datos	

Tabla 34: Tabla meta-datos ficheros

Esta tabla (Tabla 34) se irá completando con alguna otra información que se necesite para la correcta gestión de los ficheros. Además hay que tener en cuenta que los mismos nodos de ZooKeeper ofrecen otro tipo de información, como la fecha de creación del nodo o la ultima modificación, información que también podemos usar.

Para el caso de los servidores habría una tabla parecida que contendría meta-información para la gestión de los mismos, esta tabla podría tener un formato parecido al siguiente:

Dirección	
Número de ficheros	
Tipo de servidor	

Tabla 35: Tabla meta-datos servidores

Al igual que antes esta tabla (Tabla 35) puede variar y los nodos tienen la misma meta-información de la que podríamos hacernos uso también.

Para los nodos de Lista_hot y Lista_UA, los que contienen la lista de ficheros “hot” y la de los ficheros usados en el último acceso se guardará respectivamente, se guardará una estructura de List <String> con los nombres de los ficheros en cada caso.

Por ultimo tenemos el nodo de Grupos, del cual cuelgan nodos para cada grupo que tiene el usuario. Estos nodos tienen una lista como las anteriormente mencionadas, que contiene los integrantes de la lista y también quien es el dueño del grupo, entre alguna otra información que se necesitará. Una vez creado el grupo, se crean dos nodos hijos del mismo pertenecientes para los ficheros y servidores compartidos por el grupo, los cuales pueden colgar de dichos nodos o ser una lista que esta contenida en los nodos, dependiendo de la necesidad sobre la meta-información para la gestión de esta información.

Una vez determinada la estructura de nodos dentro de ZooKeeper hay que decir que nos encontramos con varias alternativas de manejar los datos o ficheros dentro de ZooKeeper.

4.4.2.1 Alternativa 1

El primero se trataría de que toda la información, ficheros, etc. se guardaran directamente en ZooKeeper, limitando el tamaño de la información al tamaño de los nodos o creando varios nodos para albergar ficheros más grandes. Obviamente con la idea de poder añadir servidores propios o de la nube, esta idea no sería viable. Podríamos considerar la posibilidad de guardar toda la información de forma replicada en ZooKeeper además de en los servidores que se solicite, pero puede ser un tanto caótica y una sobrecarga considerable para nuestro sistema, además es algo que no es tan necesaria para la idea de unificar toda la información que tiene dispersa el usuario, ya que si guardamos todo en ZooKeeper al final es algo más parecido a una nube que a unificar la información que tiene el usuario.

4.4.2.2 Alternativa 2

Otra alternativa es la de no guardar nada de información de los ficheros en los nodos. Es totalmente lo contrario a la opción anterior y se trataría de que en los nodos solo se guarde la meta-información de los ficheros, sin guardar nada más de ellos. Esta opción, obviamente, es viable con el planteamiento de nuestro servicio y sería la que menos carga al sistema produciría.

4.4.2.3 Alternativa 3

Por último tenemos la opción de guardar la meta-información de los ficheros, que queramos manejar, con la tabla de la información del fichero en el nodo y en caso de que el fichero sea de un tamaño inferior a un tamaño determinado para que todo entre en un mismo nodo, también se guarde el fichero, junto con su información de localización en el servidor.

Este método de organización sería como una mezcla de las dos anteriores, en caso de que el fichero tenga como máximo un tamaño determinado, pues el sistema se comportaría como el primer caso, con la diferencia de que el fichero también será guardado en el servidor. En caso de que el fichero sea mayor que ese tamaño predefinido, pues el sistema se comporta como el segundo caso y no guarda nada más que la información del fichero. Esto ofrece una ventaja cuando se manejan ficheros pequeños y la ventaja es doble, ya que habría por un lado una copia de seguridad del fichero que se maneje y por otro lado cuando se quiera acceder a ese fichero, tendrá un acceso más rápido, puesto que el fichero se recuperaría directamente desde ZooKeeper sin la necesidad de pedirlo al servidor pertinente.

De la misma forma este mismo sistema se puede usar para proporcionar un pequeño espacio para los usuarios, cuando no tienen asignado un servidor temporal, para que así, si un servidor en el que quieren guardar algo no esta disponible, la petición junto con el

fichero, queden guardadas de forma temporal en ZooKeeper. En este caso se aceptarían ficheros más grandes que en el anterior y en tal caso se repartirán en varios nodos.

4.4.2.4 Elección de alternativa

Primero vamos a presentar una tabla con los criterios más destacables de las alternativas, así se puede ver las alternativa que cumplen con los criterios y las que no.

Alternativas	Evita Sobrecarga del sistema	Acceso rápido a fichero	Copia de seguridad	Espacio temporal
Alternativa 1	✖	✓	✓	✓
Alternativa 2	✓	✖	✖	✖
Alternativa 3	✓ Aún que se guarde información en el servidor es de fichero que no sobrepasan los nodos, por lo que no crea sobrecarga	✓ Para los ficheros pequeño que se guardan en el nodo de la meta-información	✓ Para los ficheros pequeño que se guardan en el nodo de la meta-información	✓

Tabla 36: Comparativa alternativas implementación ZooKeeper

Como se puede ver en la tabla (Tabla 36) la **alternativa 3** es la más completa, por lo que para mi sistema considero que la, ya que en determinados casos ofrece las ventajas de la primera alternativa, de tener un acceso más rápido a la información además de tener una copia y por otro lado sigue manteniendo un orden en la jerarquía de los nodos además de que evita una sobrecarga en el sistema tanto de almacenamiento como operativa, algo que ofrece la segunda alternativa y que considero un criterio de bastante importancia. Esta elección será para la primera versión del servicio (8.3.1), pero para el primer prototipo se usará una adaptación de la **alternativa 1**, guardando toda la información en ZooKeeper, puesto que este prototipo aún no ofrecerá la posibilidad de añadir servidores.

5 Implementación e implantación

5.1 Introducción

Como se trata de un proyecto realmente ambicioso y de gran tamaño, lo primero que se debería realizar es un prototipo inicial para que ayude a evaluar algunas de las funcionalidades y limitaciones de las tecnologías elegidas y también para comprobar como se pueden ir adaptando algunas de las elecciones que se han realizado sobre el proyecto. A su vez el prototipo ayudará a comprobar realmente si el proyecto es viable o hasta que punto es viable así como si se pueden cubrir las necesidades u objetivos básicas propuestas.

Para poder probar nuestra idea primero se va a desarrollar un prototipo con funcionalidades limitadas, que servirá para determinar el posible alcance del sistema, así como probar algunos de los recursos que vamos a utilizar, como el servidor de aplicaciones (Tomcat) o el ZooKeeper. También servirá para realizar algunas pruebas básicas del funcionamiento del sistema y ser la base para seguir con el futuro desarrollo de la plataforma.

5.2 Definición del prototipo

Para definir las bases de lo que sería el primer prototipo vamos a nombrar algunas de las limitaciones del mismo:

- Para empezar el prototipo tendrá la limitación de tener un funcionamiento local, es decir tanto el servidor de aplicaciones como el modulo de ZooKeeper estarán en la misma maquina y se conectarán de forma local, además se accederá a la web de forma local.

- No se permitirá añadir servidores para los usuarios con lo que toda la información, incluido los datos, se guardará en ZooKeeper, utilizando una versión de la primera alternativa de la anteriormente descritas para la implementación el módulo de ZooKeeper (4.4.2.1).

- No permitirá crear grupos de usuarios con lo que no se podrán compartir ficheros ni servidores.

- Tampoco permite crear carpetas, por lo que todos los ficheros se guardarán tal cual.

- No se permitirá al usuario la posibilidad de cifrar los ficheros que quiera guardar.

Ahora vamos a pasar a nombrar las funcionalidades que sí se han implementado, con las respectivas limitaciones o maneras que se han hecho:

- Solo se acepta un formato alfanumérico para los nombres de los usuarios, este formato está controlado por el formulario.

- Para el formato de los ficheros además se incluye el carácter “.”.

- La gestión de la cuenta será básica, permitirá registrarse, acceder, abandonar la misma o darse de baja.

- Se ha creado un formulario de identificación de usuarios y de registro de usuarios nuevos.

- Para el registro e identificación de usuario solo se pide el nombre de usuario y contraseña. En caso de que se quiera registrar el usuario, no puede tomar un nombre ya existente, en tal caso se le avisará con lo ocurrido. En caso de que el usuario se intente identificar con un nombre de usuario o contraseña erróneos se produce un error y se le avisa al usuario del mismo.

- Para el desarrollo de la web se ha usado el servidor de aplicaciones Apache Tomcat.

.

- Se ha implementado ZooKeeper para el módulo encargado de sincronizar y centralizar la información, usando su correspondiente api para java.

- Se ha implementado para cada cliente un listado de los ficheros que contiene añadido en su cuenta particular, desde la cual directamente se pueden descargar los ficheros deseados.

- Barra el borrado de los ficheros se ha implementado un listado complementario y en paralelo al anteriormente descrito, que contiene los nombres de ficheros y checkbox’s los cuales se pueden marcar y borrar los ficheros que se han seleccionado.

- Además se ha implementado un checkbox él cual sirve para borrar todos los ficheros disponibles del cliente.

- Se ha implementado la posibilidad de abandonar la cuenta del usuario y también de darse de baja del servicio, lo cual implica el borrado, tanto de la cuenta como del contenido de la misma.

- El árbol de los nodos en ZooKeeper para este prototipo se ve bastante reducido, incluyendo solo los nodos de lo usuarios, de los cuales directamente cuelgan los ficheros.

- Se ha implementado la posibilidad de añadir notas de texto a la lista de ficheros, indicando un nombre para el mismo, además del contenido de la nota. Una vez

escrita la nota, cuando se va a guardar en el servicio se le añade internamente la extensión .txt al nombre de la nota y se guarda como un fichero.

- Si el usuario intenta tener un fichero o una nota con un nombre ya existente el sistema genera un error, avisando al usuario de lo ocurrido.

- Se ha establecido como tamaño máximo de los ficheros 500KB.

- Con los diferentes formularios se controlan las diferentes exigencias, como por ejemplo introducir un nombre y contraseña para identificarse, o seleccionar un fichero para poder subirlo al sistema. En cada caso se le avisa y guía al usuario con lo que tiene que hacer.

- Se han implementado las listas de los ficheros usados en la última sesión (los descargados y los subidos en la misma lista) y también la lista de los ficheros "hot" (contando como accesos al mismo las descargas, ya que el prototipo no permite sobrescribir un fichero ya existente).

Una vez indicadas las limitaciones del prototipo y también las funcionalidades implementadas, vamos a proceder a explicar el funcionamiento del prototipo.

Lo primero que se encuentra el usuario es un formulario para poder ingresar al sistema o en caso contrario para registrarse al mismo.

Una vez el usuario pueda ingresar en su cuenta particular se mostrará el formulario principal del prototipo donde se podrá ver una lista de los archivos que tiene ahora mismo agregados al sistema con el número total de los mismos, si es que los tiene, en cuyo caso se indicará que no tiene ficheros añadidos. Los ficheros se podrán descargar directamente desde esa lista o borrar. El formulario además permitirá al usuario añadir nuevos ficheros o notas, así como abandonar la cuenta o darse de baja, opciones anteriormente descritas.

5.3 Proceso de instalación y configuración

Puesto que el prototipo no cuenta con un instalador, hay que explicar la implantación que se tiene que realizar para poder hacerlo funcionar.

Lo primero que hay que hacer es instalar una maquina virtual de Java, es decir el JRE, ya que tanto Apache ZooKeeper como Apache Tomcat requieren una. No es estrictamente necesario instalar la última versión, de hecho la versión de Apache ZooKeeper que vamos a usar requiere como mínimo la versión 6 de JRE y la de Apache Tomcat la versión 7, pero yo recomiendo instalar la última [15], que en este caso es la versión 8 actualización 60 (8u60 ó 1.8.0_60) [16]. Descargamos la versión para Windows de las cuales, para una instalación más sencilla, se podría descargar la versión *.exe*, que es la versión con instalador.

Una vez instalado el JRE, vamos a descargar, instalar y configurar Apache ZooKeeper. Al igual que antes recomiendo la última versión estable de ZooKeeper [17], que en este caso es la versión 3.4.6. Una vez descargada, vamos a proceder a la instalar y configuración, para ayudar en este proceso se puede consultar el siguiente lugar [18] o en caso de que se prefiera una versión en video [19] [20].

Ahora vamos a proceder a hacer el mismo proceso, pero con Apache Tomcat. La ultima versión la podemos encontrar aquí [21]. Actualmente esta disponible Apache Tomcat 8 [22], más concretamente la 8.0.26. Descargamos la versión de Windows, de las cuales para una instalación más sencilla podemos descargar la versión *.exe* con su instalador. Si queremos descargar el la versión *zip*, podemos ver como instalarla en castellano el siguiente enlace [23] ó en ingles aquí [24].

Por último tenemos que arrancar el servidor de ZooKeeper, para lo cual ejecutamos el fichero "*zkServer.cmd*" en la carpeta "*bin*" dentro del directorio donde tenemos instalado ZooKeeper. Luego también tenemos que desplegar la aplicación web dentro del servidor de aplicaciones, Apache Tomcat. Para ello hay que coger el fichero *.war* del proyecto, lo copiamos en al carpeta "*webapp*" dentro del directorio donde tenemos instalado Apache Tomcat e iniciamos el servidor mediante el fichero "*startup.bat*" dentro de la carpeta "*bin*".

6 Pruebas de aceptación

Después de crear el prototipo inicial para el proyecto y realizamos algunas de las primeras pruebas de aceptación para el mismo.

6.1 Introducción

En este capitulo nos vamos a centrar en las pruebas de aceptación realzadas para la comprobación del correcto funcionamiento del prototipo, si las tecnologías elegidas son las adecuadas y poder determinar si el proyecto es viable para continuar con el desarrollo. Estas pruebas además ayudaran a validar el sistema.

6.2 Diseño de plan de pruebas

Para definir toda la batería de pruebas que se tiene que realizar con las pruebas más importantes para poder concluir como buena la aceptación del sistema vamos a utilizar el siguiente formato de tabla, donde se concluirán los detalles de las pruebas:

IDENTIFICADOR:	
DESCRIPCIÓN	
REQUISITOS	
RESULTADO ESPERADO	

Tabla 37: Diseño de la tabla que describe las pruebas de aceptación

Pasaremos a describir los campos del diseño de la tabla que describe las pruebas de aceptación (Tabla 37):

- Identificador: Cada prueba tendrá su identificador para facilitar su traza por las fases subsiguientes. El formato que se seguirá es el siguiente: PA-X
- Descripción: Descripción breve de la prueba en cuestión.
- Requisitos: Los requisitos que se ven envueltos en la prueba.
- Resultado esperado: El resultado que se espera de la prueba.

6.3 Batería de pruebas

IDENTIFICADOR: PA - 1	
DESCRIPCIÓN	Identificarse con un nombre o contraseña erróneos.
REQUISITOS	RF - 2, RF - 11, RNF - 1, RNF - 2, RNF - 3, RNF - 4, RNF - 5
RESULTADO ESPERADO	No permitir acceso e indicar el error correspondiente.

Tabla 38: Prueba de aceptación PA - 1

IDENTIFICADOR: PA - 2	
DESCRIPCIÓN	Identificarse con un nombre o contraseña correctos.
REQUISITOS	RF - 2, RF - 8, RF - 11, RNF - 1, RNF - 2, RNF - 3, RNF - 4
RESULTADO ESPERADO	Acceso a la cuenta personal del usuario, mostrándole su lista de ficheros.

Tabla 39: Prueba de aceptación PA - 2

IDENTIFICADOR: PA - 3	
DESCRIPCIÓN	Registrarse con un nombre de usuario existente.
REQUISITOS	RF - 1, RF - 11, RNF - 1, RNF - 2, RNF - 3, RNF - 4, RNF - 5
RESULTADO ESPERADO	No se produce el registro y se indica el error correspondiente.

Tabla 40: Prueba de aceptación PA - 3

IDENTIFICADOR: PA - 4	
DESCRIPCIÓN	Registrarse con un nombre de usuario nuevo.
REQUISITOS	RF - 1, RF - 8, RF - 11, RNF - 1, RNF - 2, RNF - 3, RNF - 4
RESULTADO ESPERADO	Se le permite el registro al usuario, además se accede directamente a su nueva cuenta.

Tabla 41: Prueba de aceptación PA - 4

IDENTIFICADOR: PA - 5	
DESCRIPCIÓN	Abandonar la cuenta del usuario.
REQUISITOS	RF - 3, RNF - 1, RNF - 3, RNF - 4
RESULTADO ESPERADO	El usuario abandona la cuenta y se le redirecciona al login.

Tabla 42: Prueba de aceptación PA - 5

IDENTIFICADOR: PA - 6	
DESCRIPCIÓN	Darse de baja del servicio.
REQUISITOS	RF - 4, RNF - 1, RNF - 2, RNF - 3, RNF - 4
RESULTADO ESPERADO	La cuenta del usuario será borrada y se le redirecciona a la página de login.

Tabla 43: Prueba de aceptación PA - 6

IDENTIFICADOR: PA - 7	
DESCRIPCIÓN	Agregar un fichero con un nombre ya existente.
REQUISITOS	RF - 5, RNF - 1, RNF - 2, RNF - 3, RNF - 4, RNF - 5
RESULTADO ESPERADO	Se muestra el error correspondiente.

Tabla 44: Prueba de aceptación PA - 7

IDENTIFICADOR: PA - 8	
DESCRIPCIÓN	Agregar un fichero con formato incorrecto.
REQUISITOS	RF - 5, RF - 10, RNF - 1, RNF - 3, RNF - 4, RNF - 5
RESULTADO ESPERADO	Se muestra el error correspondiente.

Tabla 45: Prueba de aceptación PA - 8

IDENTIFICADOR: PA - 9	
DESCRIPCIÓN	Agregar un fichero que exceda el tamaño

	máximo.
REQUISITOS	RF - 5, RF - 9, RNF - 1, RNF - 3, RNF - 4, RNF - 5
RESULTADO ESPERADO	Se muestra el error correspondiente.

Tabla 46: Prueba de aceptación PA - 9

IDENTIFICADOR: PA - 10	
DESCRIPCIÓN	Agregar un fichero.
REQUISITOS	RF - 5, RF - 8, RF - 9, RF - 10, RNF - 1, RNF - 2, RNF - 3, RNF - 4, RNF - 6
RESULTADO ESPERADO	El fichero se agrega a la cuenta del fichero y se actualiza su lista.

Tabla 47: Prueba de aceptación PA - 10

IDENTIFICADOR: PA - 11	
DESCRIPCIÓN	Borrar uno o varios ficheros de la lista.
REQUISITOS	RF - 7, RF - 8, RNF - 1, RNF - 2, RNF - 3, RNF - 4
RESULTADO ESPERADO	Los ficheros son borrados y se actualiza la lista.

Tabla 48: Prueba de aceptación PA - 11

IDENTIFICADOR: PA - 12	
DESCRIPCIÓN	Borrar todos los ficheros de la cuenta.
REQUISITOS	RF - 7, RF - 8, RNF - 1, RNF - 2, RNF - 3, RNF - 4
RESULTADO ESPERADO	Los ficheros son borrados y la lista queda vacía.

Tabla 49: Prueba de aceptación PA - 12

IDENTIFICADOR: PA - 13	
DESCRIPCIÓN	Descargar un fichero de la lista.
REQUISITOS	RF - 6, RNF - 1, RNF - 2, RNF - 3, RNF - 4
RESULTADO ESPERADO	El fichero se descarga en el terminal del

	usuario.

Tabla 50: Prueba de aceptación PA - 13

6.4 Análisis de resultados

Una vez echas las pruebas se comprueba si los resultados obtenidos se han ajustado a los resultados esperados o no, para ello se presenta la siguiente tabla, en la que se podrá ver cada prueba indicando con su correspondiente verificación.

PRUEBAS	RESULTADO CORRECTO
PA - 1	✓
PA - 2	✓
PA - 3	✓
PA - 4	✓
PA - 5	✓
PA - 6	✓
PA - 7	✓
PA - 8	✓
PA - 9	✓
PA - 10	✓
PA - 11	✓
PA - 12	✓
PA - 13	✓

Tabla 51: Para cada prueba se muestra si el resultado ha sido correcto.

En esta tabla (Tabla 51) podemos ver que los resultados de las pruebas de aceptación planteadas en el anterior apartado se cumplen según lo previsto.

Vamos a ver la siguiente matriz de trazabilidad para poder comprobar de forma más clara que todos los requisitos han sido probados.

	PA - 1	PA - 2	PA - 3	PA - 4	PA - 5	PA - 6	PA - 7	PA - 8	PA - 9	PA - 10	PA - 11	PA - 12	PA - 13
RF - 1			✓	✓									
RF - 2	✓	✓											
RF - 3					✓								
RF - 4						✓							
RF - 5							✓	✓	✓	✓			
RF - 6													✓
RF - 7											✓	✓	
RF - 8		✓		✓						✓	✓	✓	
RF - 9								✓		✓			
RF - 10							✓			✓			
RF - 11	✓	✓	✓	✓									
RNF - 1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
RNF - 2	✓	✓	✓	✓		✓	✓			✓	✓	✓	✓
RNF - 3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
RNF - 4	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
RNF - 5	✓		✓				✓	✓	✓				
RNF - 6										✓			

Tabla 52: Matriz de trazabilidad requisitos de software y pruebas de aceptación.

En la matriz de trazabilidad (Tabla 52) podemos ver que todos los requisitos han sido probados por al menos una prueba de aceptación. Con lo que podemos concluir que el prototipo cubre todos los requisitos de software propuestos en el análisis del problema (3.3).

7 Planificación y presupuesto

7.1 Introducción

En este capitulo vamos a ver la planificación y el presupuesto para realizar el desarrollo del proyecto y su correspondiente prototipo inicial.

7.2 Planificación

Para empezar vamos a ver la planificación de las principales tareas en realizar para el desarrollo del proyecto. Para la realización de estas tareas y sabiendo ya desde un principio que se cuenta con el tiempo justo se calcula un trabajo diario de aproximadamente 8 horas.

7.2.1 Diagrama Gantt

Las tareas con sus días aproximadas para realizar, fecha de inicio y finalización se muestran en el siguiente diagrama de Gantt:

	Task Name	Dura	Start	Finish
1	**Planteamiento del problema**	**51 days**	**Fri 01/05/15**	**Fri 10/07/15**
2	Planteamiento incial	11 days	Fri 01/05/15	Fri 15/05/15
3	Revisión planteamiento	5 days	Mon 06/07/15	Fri 10/07/15
4	**Estado del arte**	**10 days**	**Mon 10/08/15**	**Fri 21/08/15**
5	Posibles soluciones existentes	5 days	Mon 10/08/15	Fri 14/08/15
6	Tecnologías usadas	8 days	Wed 12/08/15	Fri 21/08/15
7	**Análisis del problema**	**10 days**	**Fri 14/08/15**	**Thu 27/08/15**
8	Casos de uso	3 days	Fri 14/08/15	Tue 18/08/15
9	Requisitos de software	4 days	Wed 19/08/15	Mon 24/08/15
10	**Diseño de la solución tecnica**	**15 days**	**Mon 24/08/15**	**Fri 11/09/15**
11	Diseño general	5 days	Mon 24/08/15	Fri 28/08/15
12	Propuestas de diseño	2 days	Fri 28/08/15	Mon 31/08/15
13	Desarrollo de la propuesta elegida	5 days	Tue 01/09/15	Sun 06/09/15
14	**Implementación**	**17 days**	**Fri 28/08/15**	**Sun 20/09/15**
15	Proceso de instalación y gonfiguración	6 days	Fri 28/08/15	Fri 04/09/15
16	Implementación prototipo	16 days	Tue 01/09/15	Tue 22/09/15
17	**Pruebas de aceptación**	**5 days**	**Fri 18/09/15**	**Thu 24/09/15**
18	**Planificación y documentación**	**35 days**	**Mon 10/08/15**	**Fri 25/09/15**
19	Planificación incial	2 days	Mon 10/08/15	Tue 11/08/15
20	Revisión planificación y documentación	2 days	Fri 25/09/15	Sun 27/09/15

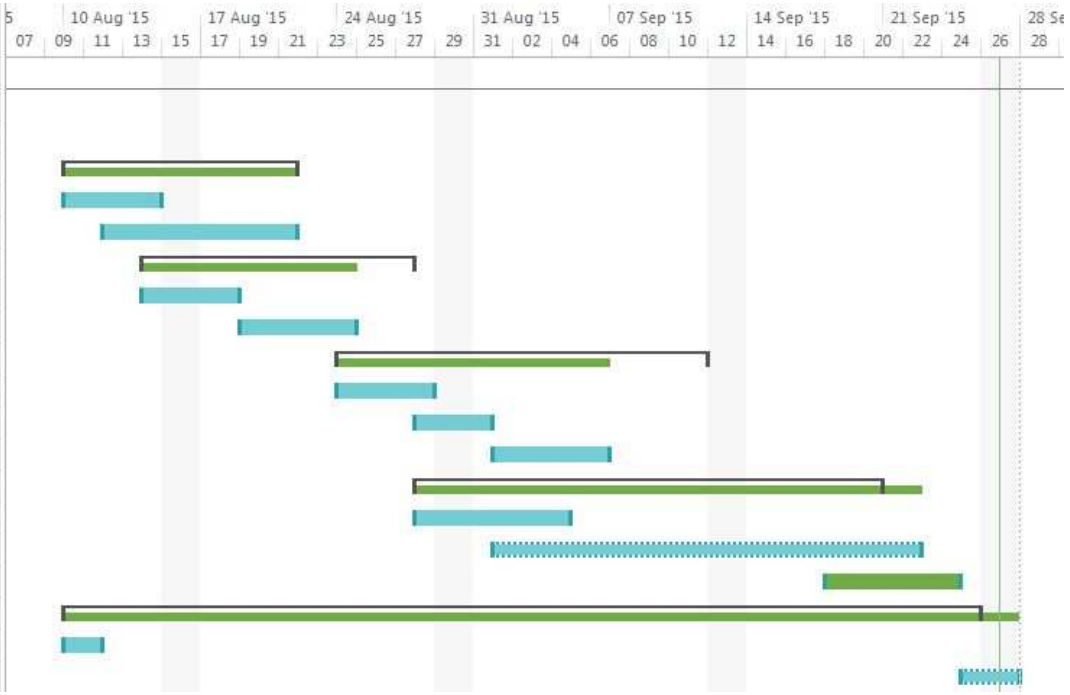


Figura 5: Diagrama Gantt de la palanificación

Podemos ver que el diagrama Gantt (Figura 5Figura 5) se compone por 7 tareas principales, de las cuales 6 tienen subtareas.

La primera tarea "*Planteamiento del problema*" no se ve en el diagrama porque se ha hecho zoom para poder ver mejor las tareas del desarrollo del proyecto, pero se trata de una tarea que tiene 2 subtareas y es básicamente la que indica el planteamiento que se ha realizado mediante reuniones, en este caso con el tutor.

La última tarea "*Planificación y documentación*" es la que se ha hecho en paralelo a todo el proyecto y trata sobre el seguimiento de la planificación, así como la realización de la documentación.

El desarrollo real del proyecto empezó el día 10/08/2015 y su estimación de finalización es el día 27/09/2015. Dado que se ha contado con un tiempo inicial de 35 días, y aún que la planificación se ha estimado con una carga diaria de trabajo de 8 horas, eso da 280 horas en total y no deja de ser un tiempo bastante justo. Por esa razón las tareas, como se puede ver, siempre se han visto solapadas y desarrolladas en paralelo, algunas de ellas en mayor medida que otras, pero sin tener en cuenta la tarea "Planificación y documentación" se puede ver que como máximo se solapan 2 tareas a la vez.

Por otro lado podemos ver como las tareas "*Análisis del problema*" y "*Diseño de la solución técnica*" se han finalizado antes del total previsto, pero la "*Implementación*" en cambio ha recibido algunos retrasos, principalmente causados por los problemas descritos en la conclusión (8.2.2).

Mientras que tareas como "*Estado del arte*" y "*Pruebas de aceptación*" si se han realizado dentro del tiempo estimado.

También se puede ver un retraso en la tarea "*Planificación y documentación*", el cual, por otro lado, al ser la tarea que delimita el camino critico de la planificación, ha ocasionado un retraso de 2 días en el desarrollo del proyecto. Esto produce que al final el proyecto se haya realizado en **37 días** y con una carga de horas aproximada de **296 horas**.

7.2.2 Ciclo de vida

Para el desarrollo del proyecto y el primer prototipo se ha tomado como referencia un **ciclo de vida en cascada**, con unas pequeñas variantes.

La adaptación del ciclo de vida en cascada que he utilizado tiene las siguientes etapas:

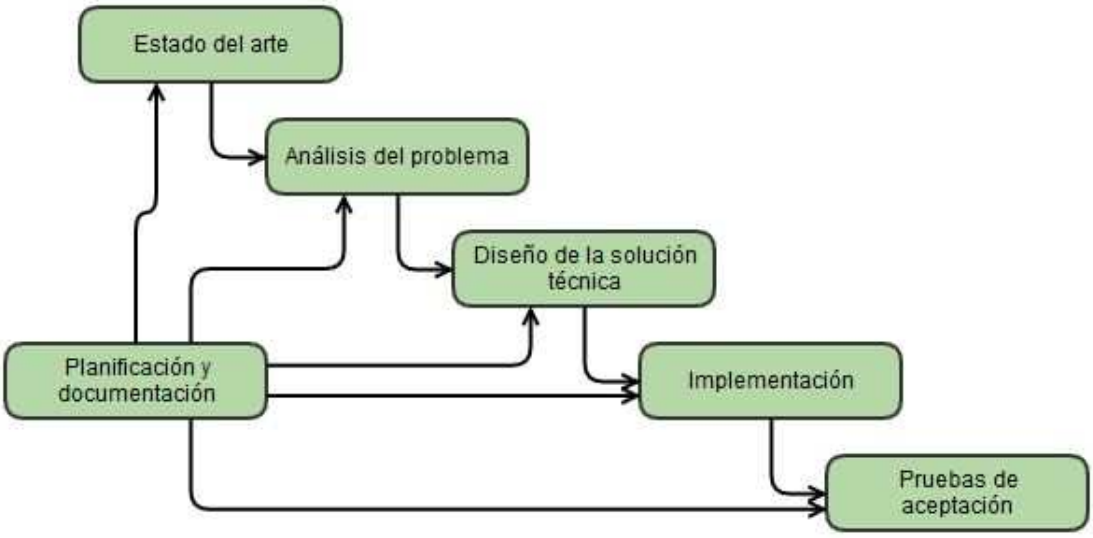


Figura 6: Ciclo de vida

Como se puede observar en la figura (Figura 6) algunas de las adaptaciones que se han realizado en el ciclo de vida en cascada, además de las tareas ha realizar, es que se ha añadido una tarea que se realiza en paralelo a todas las demás, que es la de "planificación y documentación", la cual básicamente realizar el seguimiento de la planificación, así como la documentación necesaria para el desarrollo del proyecto. Además las fases no son estrictas, en el sentido de terminar una para empezar la siguiente, si no que se llegan a solapar, como ya se vio en el diagrama de Gantt (Figura 5).

Además de este ciclo de vida en cascada hay que mencionar que se ha mezclado con el **ciclo de vida prototipado**, ya que para el desarrollo del proyecto se ha optado por empezar a realizar un prototipo inicial (**¡Error! No se encuentra el origen de la referencia.**), el cual se va a ir mejorando, hasta llegar a las fases de pruebas iniciales y la primera versión estable del servicio. Una vez finalizada la primera versión estable del servicio, se empezará también una adaptación del **ciclo de vida incremental**, puesto que se han planteado trabajos futuros para trabajar en mejoras para el servicio (8.3).

7.3 Presupuesto

Vamos a proceder a calcular el presupuesto que se ha necesitado para el desarrollo del proyecto, estimando los costes de personal a los que se les añade el resto de gastos.

7.3.1 Coste personal

Para definir el presupuesto del personal hay que definir los diferentes roles que se han necesitado para el desarrollo del proyecto. Estos roles han sido todo cubiertos por mi, con la excepción del jefe del proyecto, en el cual también ha participado el tutor.

Rol	Horas dedicadas	Coste por hora	Coste total por rol
Jefe de proyecto	40	45,00 €	1.800,00 €
Analista	60	30,00 €	1.800,00 €
Diseñador	50	35,00 €	1.750,00 €
Programador	120	20,00 €	2.400,00 €
Calidad y pruebas	25	25,00 €	625,00 €
		TOTAL	**8.375,00 €**

Tabla 53: Costes personal

Como se puede ver en la tabla (Tabla 53), los cálculos se han realizado según el coste de hora para cada rol, estimando las horas para cada rol según el diagrama de Gantt (Figura 5). Hay que tener en cuenta que la estimación de horas se ha hecho según la planificación inicial, sin tener en cuenta los 2 retrasos en las tareas "Implementación" y "Planificación y documentación". También se puede observar que el total de horas suma 295, cuando según la planificación son 280, pero esos 280 son mi carga de trabajo y como para el cálculo de los costes de personal se ha tenido en cuenta el rol de jefe de proyecto, se han añadido otras 15 horas correspondientes a las horas del tutor, ejerciendo de jefe de proyecto en tutorías.

7.3.2 Coste material

Para la realización de este coste se tiene en cuenta el hardware y el software que se ha utilizado para la realización del proyecto.

7.3.2.1 Costes hardware

En este caso se tiene en cuenta el material hardware que se ha necesitado para el desarrollo del proyecto.

Material	Precio unitario	Unidades	Periodo de amortización (años)	Nº se meses de Uso	Coste
Intel Core 2 Duo, 4GB *	0 €	1	3	1	0 €
Portátil i5, 8GB	550,00 €	1	3	0,5	7,64 €
Pendrive 32GB	20,00 €	1	3	1	0,56 €
				TOTAL	**8,19 €**

Tabla 54: Costes hardware
* Este material ya se encuentra amortizado

Como se puede observar en la tabla (Tabla 54) se considera un tiempo de amortización del material de 3 años. Además se estima el uso que se le ha dado, en meses, al material durante el proyecto.

7.3.2.2 Costes software

Al igual que para el material hardware, se hará un calculo para el material software utilizado.

Material	Precio unitario	Unidades	Periodo de amortización (años)	Nº se meses de Uso	Coste
Windows 7 profesional	135,00 €	1	3	1,16	4,35 €
Microsoft Office Professional 2003 *	0 €	1	3	1	0 €
Microsoft Project 2013 **	0 €	1	3	0,2	0 €
Apache ZooKeeper ***	0 €	1	3	1	0 €
Apache Tomcat ***	0 €	1	3	1	0 €
Eclipse Mars ***	0 €	1	3	1	0 €
JDK & JRE ***	0 €	1	3	1	0 €
				TOTAL	**4,35 €**

Tabla 55: Costes software
*** Este material ya se encuentra amortizado**
**** Usada versión de prueba**
***** Software de licencia libre**

Al igual que en el anterior cálculo de materiales, en este caso (Tabla 55) también podemos ver como se ha tomado un tiempo de amortización de 3 años y el uso se ha estimado en meses.

7.3.3 Gastos indirectos

Vamos a indicar también los gastos indirectos que se han necesitado durante la reafición del proyecto.

Concepto	Costes
Luz	40 €
Internet	70 €
Combustible	30 €
TOTAL	**140 €**

Tabla 56: Costes indirectos

En la tabla (Tabla 56) podemos ver los gastos indirectos, entre los que se encuentra el gasto del combustible que se ha necesitado para acudir a las tutorías con mi coche particular.

7.3.4 Gasto total

Para concluir vamos a terminar calculando los gastos totales necesarios para la realización del proyecto.

Concepto	Costes
Personal	8.375,00 €
Hardware	8,19 €
Software	4,35 €
Gastos indirectos	140 €
TOTAL	**8.527,54 €**

Tabla 57: Gasto total

Como se puede ver en la tabla (Tabla 57) el presupuesto total del proyecto asciende a:

8.527,54 €

Ocho mil quinientos veintisiete euros con cincuenta y cuatro céntimos.

8 Conclusiones y trabajos futuros

8.1 Introducción

Se hablará sobre las conclusiones extraídas de realizar el proyecto y el prototipo, de sus desarrollos, así como de los aspectos personales. También se verán los trabajos futuros tanto sobre el modelo como sobre el prototipo.

8.2 Conclusiones

En este apartado voy a tratar las diferentes conclusiones que he podido deducir de realizar el TFG, así como algunas de las dificultades que me he encontrado y las repercusiones que me han ocasionado.

8.2.1 De la solución propuesta

Una de las cosas más importantes de tratar es ver si se han cumplido los objetivos iniciales del proyecto. Para ello se expondrán los objetivos definidos en la introducción de este documento en una tabla y se analizarán los mismos.

Objetivos	Conseguido
Objetivo 1: Gestión de cuenta	✓ Lo propuesto para el prototipo se ha conseguido, es decir, el usuario puede crear y acceder a su cuenta personal, así como abandonarla o darla de baja. Que como una implementación futura que pueda cambiar de contraseña, entre otras.
Objetivo 2: Gestión de recursos y datos	✓ El usuario puede visualizar sus ficheros, así como añadir otros a su cuanta al igual que recuperar o borrar. La implementación de cifrado y gestión de servidores, queda para futura implementación.
Objetivo 3: Vista transparente al usuario	✓ Se le muestra al usuario una lista con todos los ficheros que tiene en su cuenta, independientemente del lugar donde se encuentren. En el caso del prototipo todos están en el mismo sitio, pero en el diseño de la solución se contempla como unificar la información cuando se implemente la gestión de servidores.

Objetivo 4: Implementación de una web	✓ Se ha creado una interfaz web simple para el prototipo, la cual se ira mejorando a medida que vayan saliendo las futuras versiones, tanto a nivel visual, como funcional.
Objetivo 5: Crear grupos de usuarios	✓ Se ha propuesto un diseño para la implementación de esta funcionalidad, pero queda pendiente su desarrollo para una futura versión del prototipo.
Objetivo 6: Prototipo inicial	✓ Se ha creado un prototipo inicial, para verificar la capacidad y funcionalidad de las tecnologías elegidas, como también implementar las primeras funcionalidades y de esta forma poder determinar una viabilidad inicial del proyecto y si se puede continuar con el desarrollo.

Tabla 58: Conclusión sobre los objetivos

Con los datos presentes en la tabla (Tabla 58), se puede ver que existe un diseño viable para cubrir las necesidades iniciales del proyecto, además de los resultados obtenidos por el prototipo. Con el análisis realizado sobre los objetivos y las pruebas realizadas sobre el prototipo, se puede concluir que el proyecto puede continuar la siguiente fase de desarrollo, ya que tiene una base sólida y una alta probabilidad de viabilidad.

8.2.2 Del proceso de desarrollo

En este apartado vamos a centrarnos sobretodo en los problemas encontrados e impacto que han tenido incluso llegando a producir alguna alternativa de planificación.

Durante el proyecto me he encontrado con varios problemas, pero los más destacables, por el esfuerzo o el tiempo invertido son los siguientes:

- Aprender ha usar ZooKeeper. Es verdad que no es un problema como tal, pero sí en su conjunto, ya que se trata de un sistema que no había usado con anterioridad y ya no solo eso, ni siquiera ningún sistema parecido, basado en nodos y clave-valor.

  Los principales problemas vinieron a la hora de la instalación y configuración. Obviamente también tuve dificultades a la hora de aprender a usar el programa o a acceder a través de su api para java, pero realmente la peor parte fue la de la instalación y configuración. Es cierto que es un proyecto bastante maduro, pero la información que hay sobre él, a parte de la documentación, tampoco es que sea especialmente abultada, con lo que a la hora de intentar buscar como instalar y configurar ZooKeeper, había mucha información y tutoriales que no eran completos, estaban obsoletos. Al principio intente configurar el sistema para que funcionará sobre Linux, y tener el servidor donde se alojaba ZooKeeper en una

distribución de ese SO, pero la información que había para instalar y configurar ZooKeeper sobre Linux era muy confusa incluso a veces incoherente, por lo que después de varios intentos y de invertir un tiempo considerable decidí realizar la configuración en Windows. Esto acabo produciendo una alternativa en la planificación que sin embargo, es cierto que a la hora del desarrollo del prototipo me vino mejor, porque tenía tanto el servidor de aplicaciones como ZooKeeper en la misma maquina y se podían comunicar en local.

- Otro problema que tuve en el desarrollo se produjo al actualizar la versión de java en mi maquina virtual. No me di cuenta de los problemas que esto podía crear. Además tenía mi anterior versión de java instalada en una dirección que no era la de por defecto, y la actualización se realizo en la dirección por defecto. En este momento el servidor de aplicaciones me empezó a dar excepciones extrañas que según parecía podían ser problemas de concordancia con las versiones de java. El problema además venía de que no tenía una instantánea muy reciente del sistema o algún backup de la maquina virtual, por eso tenía que buscar la instantánea más reciente y adecuada que tenía y recuperar todos mis avances hasta ese punto para poder continuar.

  Una vez encontrada la instantánea deseada y pasado toda la información seguía teniendo problemas y excepciones, hasta que, después de revisar anteriores versiones del proyecto, me di cuenta que en medio del caos de la versión de java había realizado copias de seguridad de los archivos del proyecto, uno a uno y se me había olvidado cambiar el direccionamiento de los servlets, que al crear copias tenían el mismo y producían una excepción. Al final no sabía si los primeros errores que me encontré eran por la actualización de java, por el error de los servlets o por las dos, que es lo más probable, pero resolver ese dilema me llevo bastante tiempo y esfuerzo, que a su vez me produjo bastante frustración y también retrasos más severos.

8.2.3 Aspectos personales

Vamos a proceder a hablar sobre los aspectos personales concluidos después de la realización del proyecto. Para ello se van a exponer las siguientes dos tablas:

Asignaturas	**Uso**
Sistemas Operativos	Manejo de ficheros y sistemas de ficheros, así como la creación de tablas para los meta-datos de los ficheros y servidores como una especia de tablas de inodos.
Diseño de Sistemas Operativos	Manejo de ficheros y sistemas de ficheros, así como la creación de tablas para los meta-datos de los ficheros y servidores como una especia de tablas de inodos.
Sistemas Distribuidos	Zookeeper, además del desarrollo del

	cliente que se instalaría en los terminales o servidores. Su implementación queda como trabajo futuro.
Tecnologías Informáticas para la Web	La creación de una página web dinámica basada en JEE.
Ingeniería de Software	Para el análisis del problema, la creación de los requisitos y los casos de uso, así como las pruebas.
Dirección de proyectos de desarrollo de software	En general para la creación de este documento.

Tabla 59: Asignaturas usadas para el desarrollo del proyecto

Tecnologías	Aprendizaje
Servicio de registro de nombres y configuración centralizada y sincronizada ZooKeeper	He tenido que aprender a instalar y configurar el servicio. También el uso de la mayoría de comandos y funcionalidades, tanto a través de la interfaz de comandos de ZooKeeper como usando la api de java.
Servidor de aplicaciones Apache Tomcat	En Tecnologías Informáticas para la Web ya usamos un servidor de aplicaciones, en concreto el GlassFish. Es cierto que una gran parte de características y uso entre los dos servidores es compartida, pero he tenido que aprender a instalar y configurar Apache Tomcat, además de algunas de sus particularidades.
Virtualización con VirtualBox	Ya había usado maquinas virtuales con VMware, pero maquinas más simples y para cosas muy concretas. En este caso opte por utilizar VirtualBox porque es totalmente libre y tiene algunas características como las instantáneas, entre otras ya incorporadas, pero a su vez también tenía que aprender a usar todas estas funcionalidades.

Tabla 60: Tecnologías aprendidas en el desarrollo del proyecto

En la primera tabla (Tabla 59) podemos ver enumeradas las asignaturas de las que se ha hecho uso para el desarrollo del prototipo (o proyecto), así como una descripción breve para indicar más específicamente los aspectos más destacados.

En la segunda tabla (Tabla 60) en cambio, podemos ver las tecnologías que he tenido que aprender a instalar/configurar y también a usar para poder plantear el proyecto y realizar el prototipo.

8.3 Trabajos futuros

En este apartado hablaré sobre algunos de los trabajos futuros que tengo pensados realizar. Estos trabajos se considerarían como los siguientes desarrollos a realizar para la mejora del prototipo y por otro lado las mejoras a la primera versión real del proyecto. Estos dos aspectos se ven reflejados en los siguientes puntos.

8.3.1 De la implementación del prototipo

En este apartado se enumerarán algunas de las opciones que se han quedado fuera del primer prototipo del sistema, pero que se tendrán que implementar para futuros prototipos y versiones de prueba hasta concluir con la primera versión estable.

- Implementar la posibilidad de que los usuarios puedan crear grupos.
- Permitir al usuario crear carpetas.
- Implementar la posibilidad de que los usuarios puedan Cifrar.
- Implementar la posibilidad de que los usuarios puedan gestionar servidores
- Mejorar la interfaz web del usuario, haciendo que la misma sea más intuitiva y clara mejorando algunas funcionalidades base de gestión y cambios en los formatos de los datos.
- Cambio de alternativa para la gestión de la información en ZooKeeper.

8.3.2 Del modelo/solución

Aquí tenemos los trabajos futuros para realizar sobre la primera versión real del proyecto, es decir las mejoras que tengo pensadas para implementar una vez creada, testeada y lanzada la primera versión del proyecto. Los trabajos futuros que he pensado son los siguientes:

8.3.2.1 A nivel de diseño y arquitectura

Estos trabajos futuros, podemos definirlos como los primordiales, puesto que se trata de las mejoras a nivel de diseño base y sobre las que ya se ha hablado algo durante el diseño del proyecto. Estos trabajos deberían empezar a implementarse en cuanto salga la primera versión oficial y estable.

- Realizar la interfaz REST del sistema, la cual se usará tanto por nuestras propias aplicaciones y medios de accesos, como por las que pueda querer diseñar algún usuario para su propia forma de acceso.

- Empezar el diseño y desarrollo de las primeras aplicaciones, empezando por las versiones para Windows y Android.

- Mejorar el cliente instalado en el servidor o terminal para que pueda avisar al sistema cuando se produzcan cambios en los ficheros accediendo directamente a los mismos desde el servidor. Esto ayudará a tener el sistema más actualizado y a no generar sobrecargas de actualización, al saber exactamente lo que se ha modificado. Estas mejoras se podrían realizar mediante FUSE o un sistema de suscripción.

8.3.2.2 Complementos al diseño y funcionalidad base

Estos trabajos futuros son los que se pueden ir implementando una vez estén los indispensables, ya que estos se consideran como complementos al sistema y que podrían ofrecer algunas funcionalidades interesantes al usuario, pero no se consideran de máxima prioridad, si no como funcionalidades complementarias.

- Calendario: Se trata de una funcionalidad que permitirá al usuario añadir “ficheros” a un calendario y poder planificarlos para un día y hora. Realmente no se añadirán los ficheros en sí, al menos no hasta que no llegue el día y la hora, es decir, se añadirán referencias de los fichero que se quieran usar y una vez que llegue el momento es cuando se buscarán los ficheros y se avisará al usuario dándole la posibilidad de descargarlos. Esto se hace para que no haya carga en el sistema ni enviar ficheros a los usuarios (en caso de usar la aplicación) antes de tiempo, ya que a lo mejor el usuario anula su petición antes del aviso programado.

  La presentación del aviso se realizará de varias formas, según el medio de petición o como esté conectado el usuario en ese momento. Si se hace a través de la web, cuando vuelva a entrar el usuario, si ya ha pasado el aviso se le avisará nada más ingresar en su cuenta o si el usuario está en ese momento identificado se le avisará en el momento para el que se ha programado. Si se cumple el momento de la programación del aviso del calendario a través de la aplicación del sistema se le enviará un aviso con la posibilidad de que el usuario pueda descargar los ficheros. Lo mismo pasaría en caso de que fuera en la aplicación móvil, se le avisaría al usuario en dicho instante. Este calendario ayudaría a los usuarios para que puedan tener sus ficheros mejor organizados y que nunca se les olviden ciertos documentos para, por ejemplo, una reunión.

- Cálculo de costes: Esta funcionalidad se basa en realizar un cálculo de costes en caso de que se quiera guardar información en la nube, es decir, realizará un cálculo de costes preventivo y orientativo, exponiendo las diferentes opciones que puede optar el usuario e indicándole la que se consideré el mejor.

- Cálculo de energía: Que la aplicación presente un cálculo de energía del sistema y ofrezca al usuario la posibilidad de definir servidores como de gran uso o de poco uso, asignando los ficheros de forma automática en el lugar más adecuado.

- Una vista por servidores: Ofrecer la posibilidad al usuario de poder visualizar su información por servidores y no toda junta. Es otra forma de visualizar los datos, es cierto que rompe en parte con la idea de la transparencia para el usuario, ya que no se le presenta toda la información junta, aún que el sistema seguirá siendo el encargado de hacer la petición al servidor mediante su dirección, pero por otro lado le ofrece una forma rápida al usuario de visualizar como tiene repartido su contenido, que pueda ver si algo está en el sitio deseado o como se han repartido las replicas, etc.

- Envío por e-mail: Se trata de que el usuario pueda enviar datos por e-mail, esto no registrará meta-información, es decir no es una capacidad de servicio tipo servidor, es simplemente otra forma que se le ofrece al usuario de compartir su información, tanto con su propio correo, como con otros.

- Cifrado y partición de ficheros: Esta es una opción de seguridad más avanzada que la de cifrar, se trata de que cuando un usuario tenga más de un servidor añadido a su cuenta, también tenga la opción de cifrar un fichero y además que el sistema se encargue de realizar una partición de dicho fichero y enviar cada parte en un servidor, de forma que solo el sistema sepa donde está cada parte y en que orden.

- Crear la posibilidad de añadir autenticación para el acceso a ciertos servidores

- Crear un servicio de nube ofrecido por mi sistema, si se ve esa necesidad por parte de los usuarios.

9 Bibliografía

[1]: https://www.dropbox.com/about

[2]: https://www.google.com/intl/es_es/drive/

[3]: https://www.jumptuit.com/

[4]: https://www.jolicloud.com/about

[5]: https://www.multcloud.com/about

[6]: https://www.synology.com/es-es/company

[7]: https://www.qnap.com/i/es/

[8]: https://zookeeper.apache.org/

[9]: http://tomcat.apache.org/

[10]: https://glassfish.java.net/

[11]: https://es.wikipedia.org/wiki/Java_EE

[12]: http://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html

[13]: http://www.boe.es/buscar/act.php?id=BOE-A-1999-23750&p=20110305&tn=1

[14]: http://www.boe.es/buscar/act.php?id=BOE-A-1996-8930

[15]: http://www.oracle.com/technetwork/java/javase/downloads/index.html

[16]: http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html

[17]: http://apache.rediris.es/zookeeper/stable/

[18]: https://ihong5.wordpress.com/2014/07/07/how-to-install-and-setup-apache-zookeeper-standalone-windows/

[19]: https://www.youtube.com/watch?v=H3QNDhIxHxE

[20]: https://www.youtube.com/watch?v=91zzlhkJzVA

[21]: http://tomcat.apache.org/whichversion.html

[22]: http://tomcat.apache.org/download-80.cgi

[23]: http://www.edu4java.com/es/servlet/servlet1.html

[24]: http://www3.ntu.edu.sg/home/ehchua/programming/howto/Tomcat_HowTo.html

10 Anexo

10.1 Ingles

10.1.1 Summary

This paper will discuss the planning and design of my TFG as well as the development of the first prototype.

It is about creating a system that is able to centralize and synchronize files from a user. It doesn't matter if the files are inside a PC, smartphone, on the cloud, you name it.

The system must have the ability to locate all the files from the account user, so that him found all the files together in the same place in a transparent way, but the real thing is that the real files are stored in different places and devices

In addition the system also needs to have functionalities that the user does not perform by hand, such as creating replicas of a file (according to specific conditions) or detect if someone edited or deleted a file from the server where it was located, among other things.

To make this project and (as with almost all), you should make a study of the competition that exists today and the alternatives or possible solutions to the problem there. The study focuses on three main alternatives, such as: cloud storage, tools or services unification of accounts and finally cloud storage NAS.

The technologies and possible uses about this study like storing data or metadata, application server, programming languages used are also studied here.

Then we could raise the analysis of use cases and requirements as well as project design, which begins with an overview of it, the alternatives and the detailed approach of the chosen alternative.

We must also mention the implementation of a first prototype, on some tests to determine the viability of the project and whether it can be carried out or if it could really cover the needs or objectives.

10.1.2 Introduction

10.1.2.1 Motivation

Throughout my degree I realized my problem was happening to me and my colleagues. This problem occurred mainly when performing practices. When we were developing practices we realized we had parts and versions of the same practice in DropBox, Google Drive, personal PC in each house in the PC university, etc. And that's the same one.

If we consider all the practices we did in all subjects, just having material dispersed by endless clouds, terminals, sometimes you were in statements or notes on smartphones. For this reason (and the interest that I have on the subject) I decided to make something that could provide all this task about where you have all the information and how to access in a simpler way to all the material that you can have in different places.
To this we must develop a platform that will manage files in a distributed way and also that I could connect to all sources of storage that the user were to use. And for any type of user, both casual and professional.

The interest in this topic comes to build something different from what is proposed today one of the fastest growing systems in terms of data storage, which is cloud storage. This type of storage has its drawbacks, but as well offers many good things, so the intention is to seek middle ground between the benefits offered by the solution storage and its shortcomings. The idea is to create a storage system vitaminized, so in addition to managing cloud storage, management also provide a source of user's own storage, in a kind of fusion between the two types of storage, personal cloud and cloud .

The project, largely based on the following two concepts:

- Allow user / client to centralize and unify all sources of storage, allowing you to add own servers or terminals in a transparent manner

- And on the other hand allow the system to be active rather than passive, offering guests an intelligence based primarily on defining rules and other functions.

What does it mean to centralize and unify?

The aim is that the user can use our system or as a cloud storage or that the user can add to your account and associate other types of storage, whether other cloud storage accounts as Dropbox and Google Drive, your own servers or terminals such as PC's, smartphones, etc; and have all this centralized and transparent information for the user. So that the user requests a file system and the same system is responsible for knowing where to look for the file and give it to the user. In the same way the system should be responsible for synchronizing and updating information in case there is any change in any of the files directly from the associated servers, of which it is aware.

Is it an active system?

The idea is that the system performs actions transparently to the user in managing the data, and the user can manage and create their own rules and schedule to everyone's taste. For example, if a file file becomes "hot" (accessible only a certain number of times) that it becomes an important file and automatically make several replicas of the same servers that the user has or on the same system, depending on how the rule is defined. The definition of rules is an interesting aspect that makes our system to be active and do things for us, but in a complementary way the system must also have the ability, as mentioned above, to update your information if it occurs some changes in files or perform some activities that are not available to user management

10.1.3 Objetives

We shall define the objectives that have to have our system so it can be useful to our users

- The user must have the ability to have your own and to manage it, at least with the most basic actions, such as access to the account, leave it, close it. In future versions of the prototype it will also offer the possibility of changing the same information, such as your password (this option is not available in the prototype (5.2)).

- Besides that the user has to be able to add, delete or retrieve and display all the files you have on your account. Both the option of encrypt, manage servers as you might have in your account, would be pending and future work (this option is not available in the prototype (5.2)).

- The service has to offer a clear view to the user so that the user does not have to worry about where each file, the system should handle the location and the request of it and offer it to the user.

- Users must use the web to access, but later versions may also use the application or the REST interface so they can create their own form of access and system management (10.2.3.2.1).

- Users need to be able to have the future possibility (10.2.3.1) to create groups of users, who can share files, with the permission of the owner, or entire servers with access to the information they contain.

- Create an initial prototype by which we can begin to use the system and check its initial scope, basic operation and check its viability.

10.1.4 Memory Structure

In this section we will proceed to describe the structure of memory and orientation to help the reader to follow good reading it. We are going to describe the contents of each main chapter and annexes concisely.

- **Chapter 1:** It is the chapter in which we live and which serves as an introduction to both the project and the document

- **Chapter 2:** This chapter is where you can see what the competition offers related to my project, and some possible solutions that could bring the issue that concerns me, to finish linking all the options together . In addition I will also study related technologies to complete the work, some of the possible existing alternatives in each case and the relevant comparative between them.

- **Chapter 3:** This chapter is dedicated to analysis of the problem, to do a study of requirements it will be made by exposing them in tables. Also a study of cases of possible uses for the system will also be made to complete relating the requirements and use cases in a traceability matrix.

- **Chapter 4:** Here we see one of the most compelling chapter, related to the project design. For starters, as a way of introducing a general design of the project it will take, and then continue with design alternatives that arise. Once the study of alternatives made and justified the choice of one of them, we will proceed to develop in detail the chosen design.

- **Chapter 5:** After performing the design we will proceed to define the implementation and deployment of the system. To do a class diagram of the prototype created and a study of the same high level as well as the definition of the same it will be done. Besides that we will indicate where to download the necessary software and its corresponding configuration to implement the system.

- **Chapter 6:** In this chapter a study of the results of the prototype will be made as well as the feasibility of the project. For this a "test suite" of acceptance and the "analysis" of the same performed tests.

- **Chapter 7:** In this chapter we will see the planning done for the project, and the budget necessary for the development of analysis, design, prototype, etc.

- **Chapter 8:** Here you will discuss the lessons learned from implementing the project and the prototype, development, and personal aspects. Further work on the model as on the prototype will also be discussed.

- **Chapter 9:** This chapter contains the bibliography compiled for the development and study of the project.

- **Annex 1 (10.1)**: This is the appendix containing the required content in English, which includes the introduction, findings, and a summary of the document.

10.2 Conclusions and Future Work

10.2.1 Introduction

It will discuss conclusions drawn from implementing the project and the prototype, its developments, as well as personal aspects. Further work on both the model and in the prototype also will.

10.2.2 Conclusions

In this section I will try the different conclusions that I could deduct performing the TFG, as well as some of the difficulties that I have encountered and the impact they have caused me.

10.2.2.1 Of the proposed solution

One of the most important things to try is to see if they have fulfilled the initial objectives of the project. To this end the objectives set will be presented in the introduction to this document on a table and the same will be analyzed.

Objetives	Got
Objetive 1: Gestión de cuenta	✓ The proposed prototype is achieved, the user can create and access his personal account, as well as leave or shoot it down. As a future implementation that can change his password, among other.
Objetive 2: Gestión de recursos y datos	✓ The user can view your files, and add, download or delete. Implementing encryption and server management, is for future implementation.
Objetive 3: Vista transparente al usuario	✓ It shows the user a list of all files, regardless of where they are. In the case of the prototype are all in the same place, but in the design of the solution is seen as unifying information when server management is implemented.
Objetive 4: Implementación de una web	✓ It has created a simple web interface for the prototype, which will be improved as they become future versions, both visually, as functional.
Objetive 5: Crear grupos de usuarios	✓ I have proposed a design for the

	implementation of this functionality, but its development is still pending for a future version of the prototype.
Objetive 6: Prototipo inicial	✓ I have created an initial prototype to verify the capacity and functionality of the chosen technologies, as well as implement the first functions to determine an initial viability of the project and whether it can continue development.

Table 61: Conclusion on objectives

With the data in the table (Table 61), you can see that there is a viable design to meet the initial needs of the project, and the results obtained by the prototype. With the analysis of the objectives and tests on the prototype, it can be concluded that the project can continue the next phase of development, as it has a solid base and a high probability of viability.

10.2.2.2 Development process

In this section we will focus especially on the problems encountered and the impact they have had even going to produce some alternative planning.

During the project I have encountered several problems, but the most noteworthy for the effort and time invested are:

- Learning to use ZooKeeper. It is certainly not a problem as such, but as a whole, as it is a system that had not used before, and not only that, not even any similar system, based on nodes and key-value.

  The main problems came when the installation and configuration. Obviously I also had difficulty learning to use the program or access through its api for java, but really the worst part was the installation and configuration. True, it is a fairly mature project, but the information there is about it, apart from the documentation, nor is it particularly large, so when trying to find how to install and configure ZooKeeper, there was a lot of information and tutorials they were not complete, they were obsolete. At first I tried configuring the system so that it will run on Linux, and have the server where ZooKeeper lodged in a distribution of that OS, but the information I found to install and configure ZooKeeper on Linux was very confusing sometimes even incoherent, so after several attempts and invest considerable time I decided to make settings in Windows. This ended up producing a change in planning however, it is true that when the development of the prototype came to me better, because I had both ZooKeeper server applications on the same machine and could communicate locally.

- Another problem I had in the development came to update my version of Java virtual machine. I did not realize the problems it could create. Also had my older version of java installed in a direction that was not the default, and the update

was in the default address. At this point the application server started giving me that it seemed strange exceptions could be correctly matched with the versions of java. The problem also came from that was not a very recent snapshot of the system or a backup of the virtual machine, so he had to find the most recent snapshot that was appropriate and recover all my progress up to that point to continue.

Once you have found the desired instant and passed all the information still had problems and exceptions, after reviewing earlier versions of the project, I realized that in the chaos of the version of java, there were backup project files one by one and I forgot to change the routing of servlets, and to create they kept copies and produced the same exception. In the end I did not know if the first mistakes I encountered were by updating java, the error of the servlet or both, which is most likely, but solve this dilemma I take considerable time and effort, which in turn I produced a lot of frustration and also more severe delays.

10.2.2.3 Personal aspects

We will proceed to talk about personal issues concluded after the completion of the project. To do so will make the following two tables:

Subjects	Use
Operating systems	Management of files and file systems, as well as creating tables for meta-data of the files and servers as a kind of inode tables.
Design of Operating Systems	Management of files and file systems, as well as creating tables for meta-data of the files and servers as a kind of inode tables.
Distributed Systems	Zookeeper, besides the development of the client to be installed on terminals or servers. Its implementation is as future work.
Information Technology for the Web	Creating a dynamic website based on JEE.
Software engineering	For the analysis of the problem, building requirements and use cases, as well as testing.
Project Management Software Development	Overall the creation of this document.

Table 62: Courses used for the project

Technologies	Learning
Name registration service and centralized configuration and synchronized	I had to learn how to install and configure the service. Also the use of most

ZooKeeper	commands and functions, both through the command interface ZooKeeper as using the Java API.
Apache Tomcat Application Server	We already use an application server, specifically GlassFish. It is true that a lot of features and usage between the two servers is shared, but I had to learn how to install and configure Apache Tomcat, along with some of their particularities.
VirtualBox virtualization	Had already used VMware, but more simple machines. In this case chooses to use VirtualBox because it is completely free and has some features such as snapshots, among others already built, but also had to learn how to use all these features.

Table 63: Tecnologías aprendidas en el desarrollo del proyecto

In the first table (Table 62) we can see listed the subjects that has been used for prototype development (or project), and a brief description to indicate more specifically the highlights.

In the second table (Table 63) Instead, we can see the technologies I've had to learn to install / configure and use to raise the project and make the prototype.

10.2.3 Future Work

In this section I will discuss some of the future work that I have designed. These works would be considered to perform the following developments to improve the prototype and secondly improvements to the first actual draft. These two aspects are reflected in the following points.

10.2.3.1 Prototype implementation

This section will list some of the options that have been left out of the first prototype of the system, but that will have to implement for future prototypes and trial versions until concluding with the first stable version

- Implement the possibility that users can create groups.
- Allow user to create folders.
- Implement the possibility that users can encrypt.
- Implement the possibility that users can management servers

- Improve the web user interface, making it more intuitive and clear improving some functionalities based management and changes in data formats.

- Exchange alternative for the management of information in ZooKeeper.

10.2.3.2 Model / solution

Here is the future for the first real version of the project work, improvements that I have intended to implement once created, tested and launched the first version of the project. Future work that I thought are:

10.2.3.2.1 In terms of design and architecture

These future work, we can define them as the primary, since it is the improvements at the level of "basic design" and which has already been mentioned something during the desing of project. This work should start to be implemented when it comes out the first official stable version.

- REST interface make the system, which will be used by our own applications and means of access to both, for which a user may want to design their own form of access.

- Begin the design and development of the first applications, starting with versions for Windows and Android.

- Improve client installed on the server or terminal so you can tell the system when changes in files directly accessing them from the server. This will help you have the most current system does not generate update surges, knowing exactly what has changed. These improvements could be made by FUSE or a subscription system.

10.2.3.2.2 Complements the design and database functionality

These are the future work that may be implemented once they are indispensable, as these are considered as supplements to the system and could offer some interesting user features, but are not considered a high priority, but as complementary functionalities.

- Calendar: This is a feature that allows the user to add "files" to a calendar to schedule them a day and time. Actually no files are added itself, at least not until you get the date and time, ie, references file you want to use and once the time comes be added is when the files are searched and it will warn the user given the opportunity to download them. This is done so that no load on the system or send files to users (in case of using the application) early, because maybe the user cancels the request before the scheduled announcement.

The presentation of the warning is done in several ways, according to the request or as the user is currently connected. If done through the web, when reentering the user, if it has passed the notice will be notified no more logging into your account, or if the user is in the identified time will be advised at the time for which the announcement is scheduled. If the time of programming schedule announcement through the application of the system is met, the system will send a notice to the user with the possibility to download the files. The same would happen if it was in the mobile application, it will warn the user at that instant. This calendar would help users so they can have their files better organized and that they will never forget certain documents, for example a meeting.

- Costing: This functionality is based on a calculation of costs in case you want to save information in the cloud, that is, perform a "cost estimate" preventive and indicative, explaining the different options you can choose the user and telling him which is the best one.

- Energy calculation: That the application present a calculation of system power and provides the user the ability to define servers and high use or little use, assigning files automatically in the right place.

- A view of servers: Provide the user the possibility to display their information servers and not all together. Another way to visualize data, it is true that breaks in part with the idea of transparency for the user, since it is not presented with all the information together, yet the system will remain in charge of making the request to the server via his direction but otherwise offers the user a quick way to visualize how their content is distributed, peritiendo the user can see if something is in the desired location or have distributed replicas, etc.

- Emailing: This is the user to send data by e-mail, it will not record meta-information, ie it is not a style capability server service is just another way that it offers the user to share information both with their own address, as with others.

- File encryption and partition: This is a more advanced security option that encrypt, it is that when a user has more than one server added to your account also have the option to encrypt a file and also the system responsible for making a partition of this file and send each part on a server, so that only the system knows where each part is and in what order.

- Create the possibility of adding authentication for access to certain servers

- Create a cloud service offered by my system, if it is this need by users.